Image classification project for the course Introduction to Deep Learning

Imports and loading data

```
# Importing needed libraries here

import time
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns
import keras
from keras import layers, models, optimizers, regularizers
from keras.layers import Dense, Dropout, BatchNormalization
from keras.datasets import cifar100
from keras.models import Sequential
from keras.callbacks import EarlyStopping, LearningRateScheduler,
ReduceLROnPlateau
from tensorflow.keras.optimizers.schedules import ExponentialDecay
from sklearn.metrics import confusion_matrix, accuracy_score

# CIFAR-100 class names
# Source: https://www.geeksforgeeks.org/cifar-100-dataset/
class_names = [
    'apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee',
'beetle', 'bicycle', 'bottle',
    'bowl', 'boy', 'bridge', 'bus', 'butterfly', 'camel', 'can',
'castle', 'caterpillar', 'cattle',
    'chair', 'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch',
'crab', 'crocodile', 'cup', 'dinosaur',
    'dolphin', 'elephant', 'flatfish', 'forest', 'fox', 'girl',
'hamster', 'house', 'kangaroo', 'keyboard',
    'lamp', 'lawn_mower', 'leopard', 'lion', 'lizard', 'lobster',
'man', 'maple_tree', 'motorcycle', 'mountain',
    'mouse', 'mushroom', 'oak_tree', 'orange', 'orchid', 'otter',
'palm_tree', 'pear', 'pickup_truck', 'pine_tree',
    'plain', 'plate', 'poppy', 'porcupine', 'possum', 'rabbit',
'raccoon', 'ray', 'road', 'rocket',
    'rose', 'sea', 'seal', 'shark', 'shrew', 'skunk', 'skyscraper',
'snail', 'snake', 'spider',
    'squirrel', 'streetcar', 'sunflower', 'sweet_pepper', 'table',
'tank', 'telephone', 'television', 'tiger', 'tractor',
    'train', 'trout', 'tulip', 'turtle', 'wardrobe', 'whale',
'willow_tree', 'wolf', 'woman', 'worm'
]
```

```
# Load CIFAR-100 dataset
(cifar100_train, cifar100_test) =
tf.keras.datasets.cifar100.load_data()

(x_train, y_train) = cifar100_train
(x_test, y_test) = cifar100_test
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-
python.tar.gz
169001437/169001437 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```

```
# Using 10 classes and 200 images per class
classes = [9, 19, 29, 39, 49, 59, 69, 79, 89, 99]
num_images = 200

# Names of the selected classes
selected_class_names = [class_names[i] for i in classes]

# Create the test and training sets
x_train_selected, y_train_selected = [], []
x_test_selected, y_test_selected = [], []

for class_id in classes:
    train_indices = np.where(y_train.flatten() == class_id)[0]
[:num_images]
    test_indices = np.where(y_test.flatten() == class_id)[0]
[:num_images // 5]

    x_train_selected.append(x_train[train_indices])
    y_train_selected.append(y_train[train_indices])
    x_test_selected.append(x_test[test_indices])
    y_test_selected.append(y_test[test_indices])

# Convert lists to NumPy arrays
x_train_selected = np.concatenate(x_train_selected, axis=0)
y_train_selected = np.concatenate(y_train_selected, axis=0)
x_test_selected = np.concatenate(x_test_selected, axis=0)
y_test_selected = np.concatenate(y_test_selected, axis=0)

# Create a mapping from original class labels to new labels (0 to 9)
class_mapping = {class_id: i for i, class_id in enumerate(classes)}

# Map original class labels to new labels
y_train_selected = np.vectorize(class_mapping.get)(y_train_selected)
y_test_selected = np.vectorize(class_mapping.get)(y_test_selected)

# One-hot encode the labels
y_train_selected = tf.keras.utils.to_categorical(y_train_selected,
len(classes))
y_test_selected = tf.keras.utils.to_categorical(y_test_selected,
len(classes))
```

```
# Normalize the image data
x_train_selected = x_train_selected.astype("float32") / 255
x_test_selected = x_test_selected.astype("float32") / 255

# Display one image from each class
fig, axes = plt.subplots(1, len(classes), figsize=(15, 3))

for i, class_id in enumerate(classes):
    index = np.where(y_train.flatten() == class_id)[0][0]  # Select
first occurrence
    image = x_train[index]

    axes[i].imshow(image)
    axes[i].set_title(class_names[class_id])
    axes[i].axis("off")

plt.show()
```


```
print(selected_class_names)
```

```
['bottle', 'cattle', 'dinosaur', 'keyboard', 'mountain', 'pine_tree',
'rocket', 'spider', 'tractor', 'worm']
```

Function for the CNN

```
# Function for building the CNN (with preset hyper parameters)

def cnn(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        optimizer_name='adam',
        learning_rate=0.001,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):

    # Initialize model
    model = Sequential()
```

```
    # Loop for Convolutional layers
    # When calling with conv_layers=3, it has three stacks so total of
6 convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(0.2))
        filters *= 2

    # Flatten and Dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        optimizer = optimizers.SGD(learning_rate=learning_rate)

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

    # Early stopping (starting from 5th epoch for warm-up)
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])
```

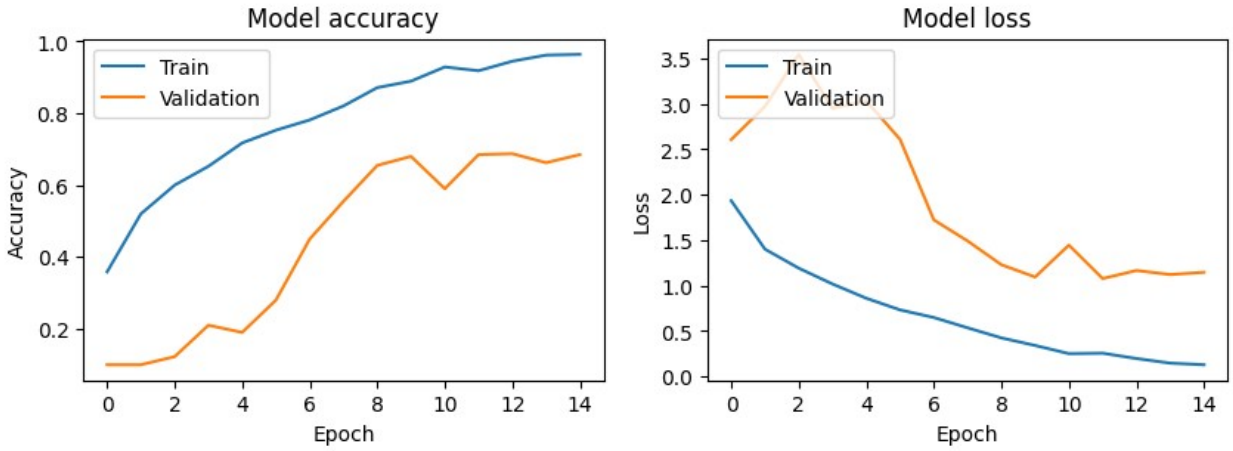
```
    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history
```

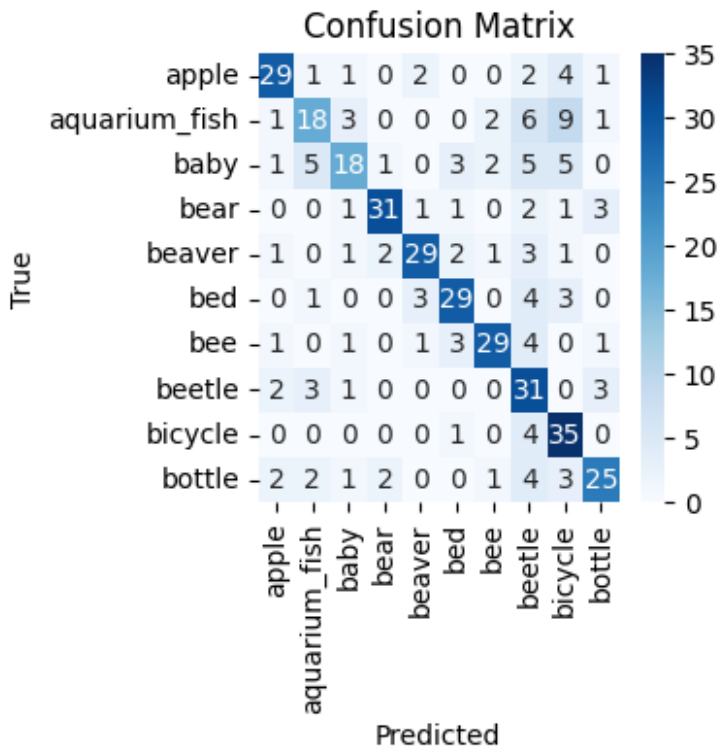
Test with default parameters

```
# Test with default parameters to get a baseline score
model_default = cnn()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 28s 325ms/step - accuracy: 0.2623 - loss:
2.2721 - val_accuracy: 0.1000 - val_loss: 2.6058
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 308ms/step - accuracy: 0.5166 - loss:
1.4378 - val_accuracy: 0.1000 - val_loss: 2.9711
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 326ms/step - accuracy: 0.5970 - loss:
1.2159 - val_accuracy: 0.1225 - val_loss: 3.5428
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 309ms/step - accuracy: 0.6571 - loss:
1.0094 - val_accuracy: 0.2100 - val_loss: 2.9494
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 336ms/step - accuracy: 0.7289 - loss:
0.8341 - val_accuracy: 0.1900 - val_loss: 3.0328
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 303ms/step - accuracy: 0.7634 - loss:
0.7023 - val_accuracy: 0.2800 - val_loss: 2.6087
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 327ms/step - accuracy: 0.8034 - loss:
0.5985 - val_accuracy: 0.4500 - val_loss: 1.7234
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 43s 352ms/step - accuracy: 0.7974 - loss:
0.5690 - val_accuracy: 0.5550 - val_loss: 1.4915
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 308ms/step - accuracy: 0.8778 - loss:
0.4093 - val_accuracy: 0.6550 - val_loss: 1.2304
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 312ms/step - accuracy: 0.8934 - loss:
0.3282 - val_accuracy: 0.6800 - val_loss: 1.0953
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 300ms/step - accuracy: 0.9373 - loss:
0.2364 - val_accuracy: 0.5900 - val_loss: 1.4458
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 318ms/step - accuracy: 0.9275 - loss:
0.2381 - val_accuracy: 0.6850 - val_loss: 1.0781
Epoch 13/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 307ms/step - accuracy: 0.9441 - loss:
0.1984 - val_accuracy: 0.6875 - val_loss: 1.1667
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 326ms/step - accuracy: 0.9666 - loss:
0.1445 - val_accuracy: 0.6625 - val_loss: 1.1223
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 305ms/step - accuracy: 0.9719 - loss:
0.1189 - val_accuracy: 0.6850 - val_loss: 1.1472
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 114ms/step - accuracy: 0.6404 - loss:
1.2642
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 80ms/step
Total training time: 393.46 seconds
Test accuracy: 0.6850
Test loss: 1.0781
```





Baseline scores

Total training time: 393.46 seconds Test accuracy: 0.6850 Test loss: 1.0781

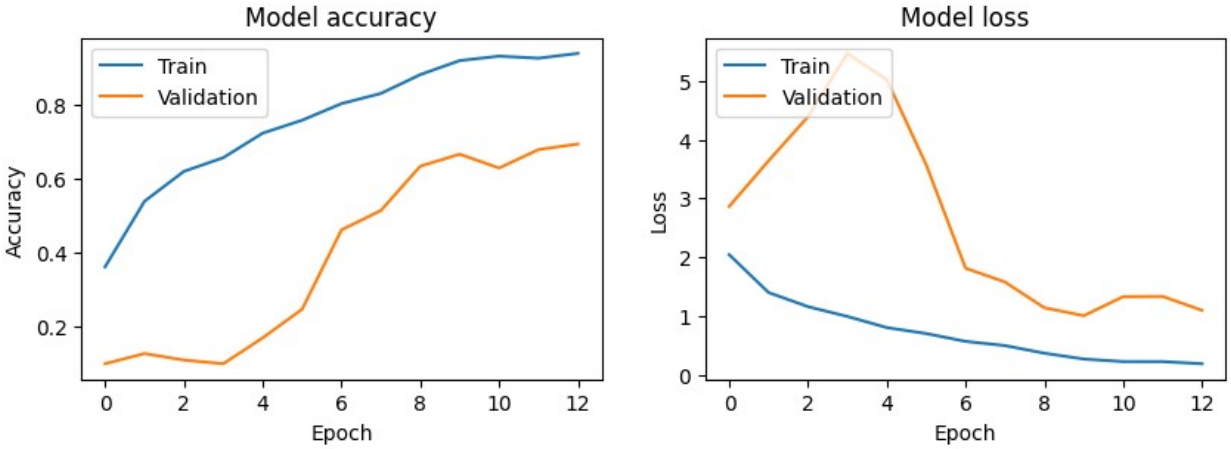
Experiments

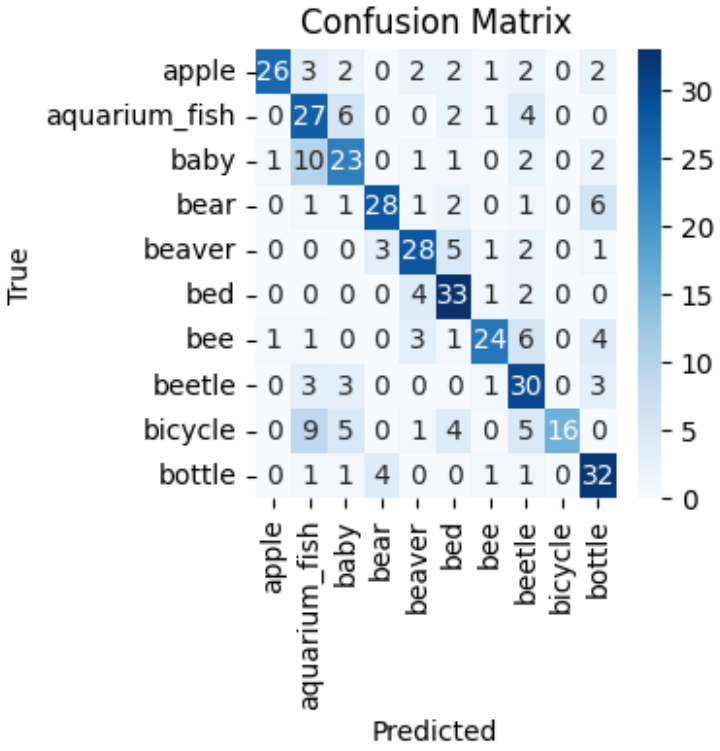
Experiment 1 (dense layer depth)

```
# Experiment 1: Increase dense layer depth
cnn(dense_units=256)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 25s 291ms/step - accuracy: 0.2801 - loss:
2.3577 - val_accuracy: 0.1000 - val_loss: 2.8640
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 267ms/step - accuracy: 0.5341 - loss:
1.4176 - val_accuracy: 0.1275 - val_loss: 3.6400
Epoch 3/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 274ms/step - accuracy: 0.6284 - loss:
1.1316 - val_accuracy: 0.1100 - val_loss: 4.3878
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 269ms/step - accuracy: 0.6645 - loss:
0.9695 - val_accuracy: 0.1000 - val_loss: 5.4638
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 276ms/step - accuracy: 0.7300 - loss:
0.7881 - val_accuracy: 0.1700 - val_loss: 5.0165
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 275ms/step - accuracy: 0.7868 - loss:
0.6589 - val_accuracy: 0.2475 - val_loss: 3.5725
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 258ms/step - accuracy: 0.8145 - loss:
0.5792 - val_accuracy: 0.4625 - val_loss: 1.8141
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 287ms/step - accuracy: 0.8350 - loss:
0.5146 - val_accuracy: 0.5150 - val_loss: 1.5795
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 267ms/step - accuracy: 0.8843 - loss:
0.3471 - val_accuracy: 0.6350 - val_loss: 1.1418
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 291ms/step - accuracy: 0.9220 - loss:
0.2633 - val_accuracy: 0.6675 - val_loss: 1.0077
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 271ms/step - accuracy: 0.9403 - loss:
0.2114 - val_accuracy: 0.6300 - val_loss: 1.3303
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 273ms/step - accuracy: 0.9137 - loss:
0.2511 - val_accuracy: 0.6800 - val_loss: 1.3355
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.9414 - loss:
0.1926 - val_accuracy: 0.6950 - val_loss: 1.0987
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step - accuracy: 0.6624 - loss:
0.9902
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 73ms/step
Total training time: 263.75 seconds
Test accuracy: 0.6675
Test loss: 1.0077
```






```
<keras.src.callbacks.history.History at 0x7a1bab438790>
```

Experiment 1, increased dense layer depth to 256

Total training time: 263.75 seconds Test accuracy: 0.6675 Test loss: 1.0077

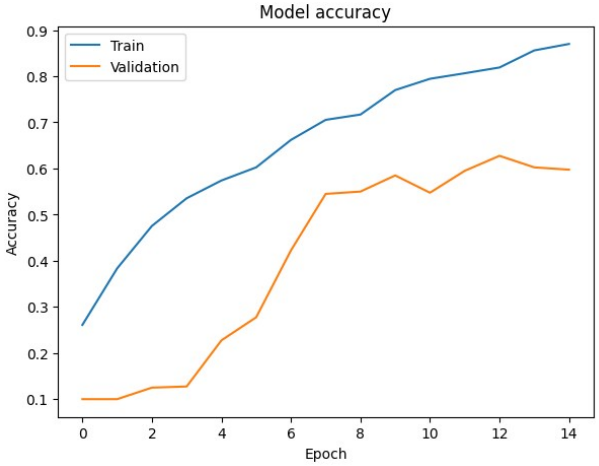
Faster, worse accuracy, better loss

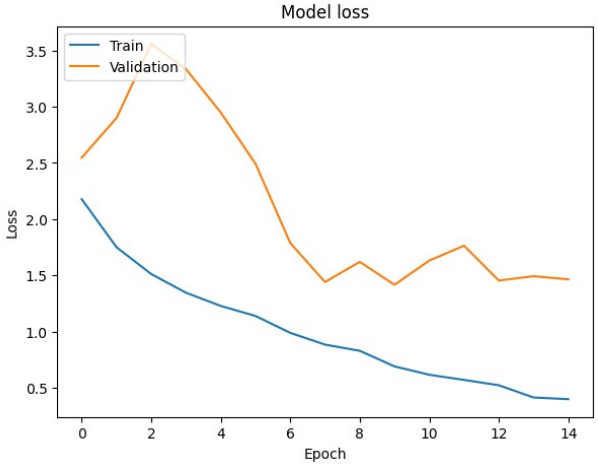
Experiment 2 (convolutional layers)

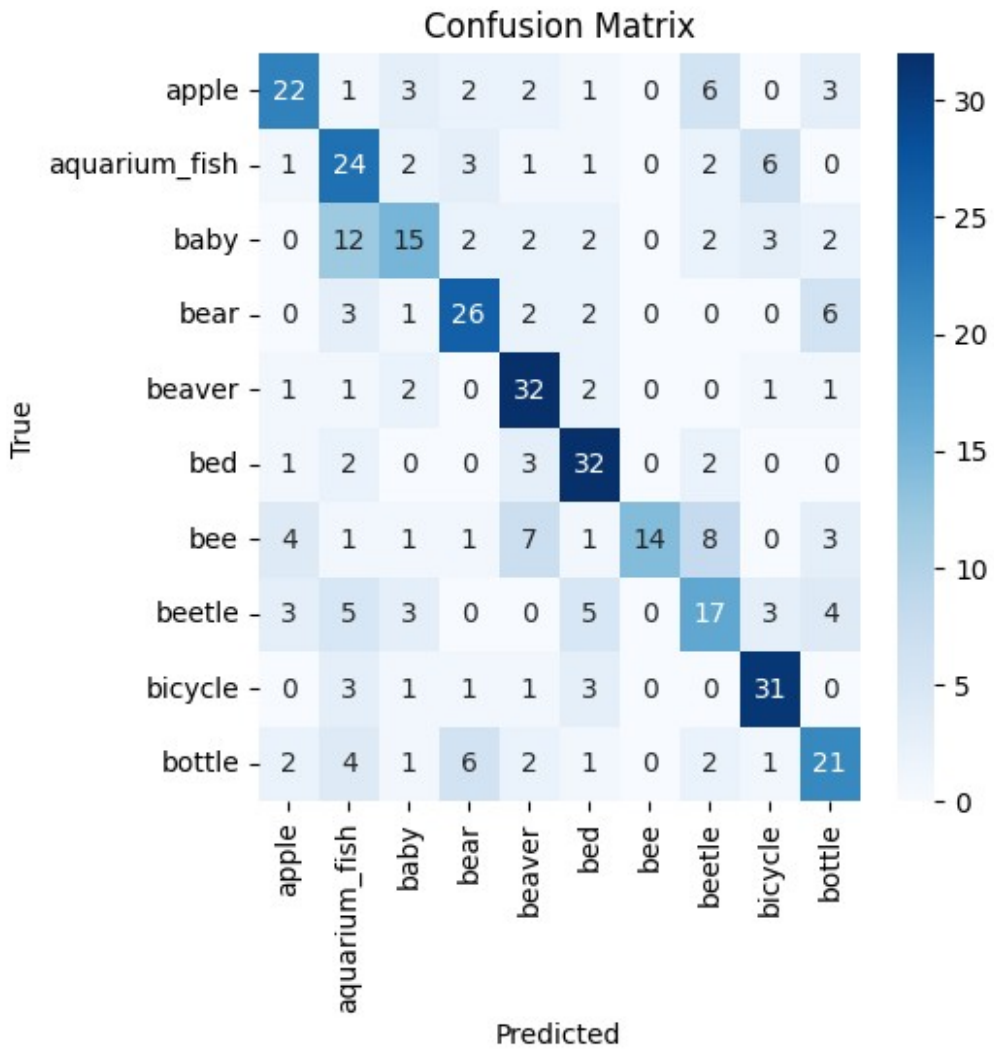
```
# Experiment 2: More convolutional layers
cnn(conv_layers=5)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 99s 1s/step - accuracy: 0.1973 - loss:
2.4769 - val_accuracy: 0.1000 - val_loss: 2.5475
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 141s 1s/step - accuracy: 0.3747 - loss:
1.7856 - val_accuracy: 0.1000 - val_loss: 2.8991
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 138s 1s/step - accuracy: 0.4823 - loss:
1.5052 - val_accuracy: 0.1250 - val_loss: 3.5572
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 144s 1s/step - accuracy: 0.5521 - loss:
1.3209 - val_accuracy: 0.1275 - val_loss: 3.3334
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 142s 1s/step - accuracy: 0.5692 - loss:
1.2440 - val_accuracy: 0.2275 - val_loss: 2.9487
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 143s 1s/step - accuracy: 0.6067 - loss:
1.1643 - val_accuracy: 0.2775 - val_loss: 2.4934
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 143s 1s/step - accuracy: 0.6651 - loss:
0.9888 - val_accuracy: 0.4225 - val_loss: 1.7887
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 140s 1s/step - accuracy: 0.7114 - loss:
0.8656 - val_accuracy: 0.5450 - val_loss: 1.4422
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 146s 1s/step - accuracy: 0.7368 - loss:
0.7763 - val_accuracy: 0.5500 - val_loss: 1.6210
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 138s 1s/step - accuracy: 0.7879 - loss:
0.6456 - val_accuracy: 0.5850 - val_loss: 1.4178
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 139s 1s/step - accuracy: 0.8178 - loss:
0.5525 - val_accuracy: 0.5475 - val_loss: 1.6327
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 137s 1s/step - accuracy: 0.8250 - loss:
0.5225 - val_accuracy: 0.5950 - val_loss: 1.7651
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 83s 1s/step - accuracy: 0.8167 - loss:
0.5439 - val_accuracy: 0.6275 - val_loss: 1.4551
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 83s 1s/step - accuracy: 0.8569 - loss:
0.3980 - val_accuracy: 0.6025 - val_loss: 1.4937
```

```
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.8758 - loss:
0.3617 - val_accuracy: 0.5975 - val_loss: 1.4660
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 94ms/step - accuracy: 0.5772 - loss:
1.4596
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 121ms/step
Total training time: 1960.59 seconds
Test accuracy: 0.5850
Test loss: 1.4178
```







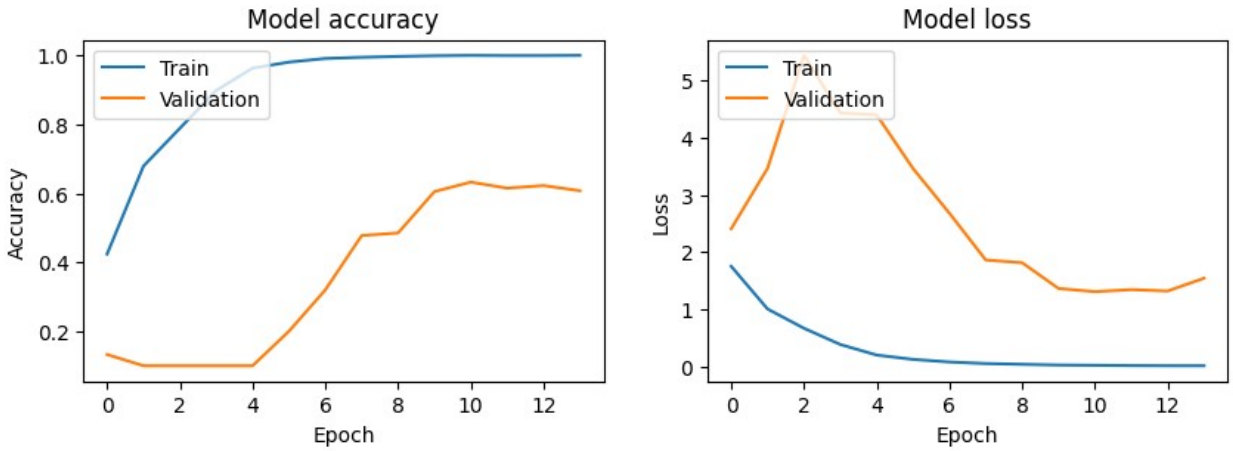
Experiment 2, increased convolutional layers to 5

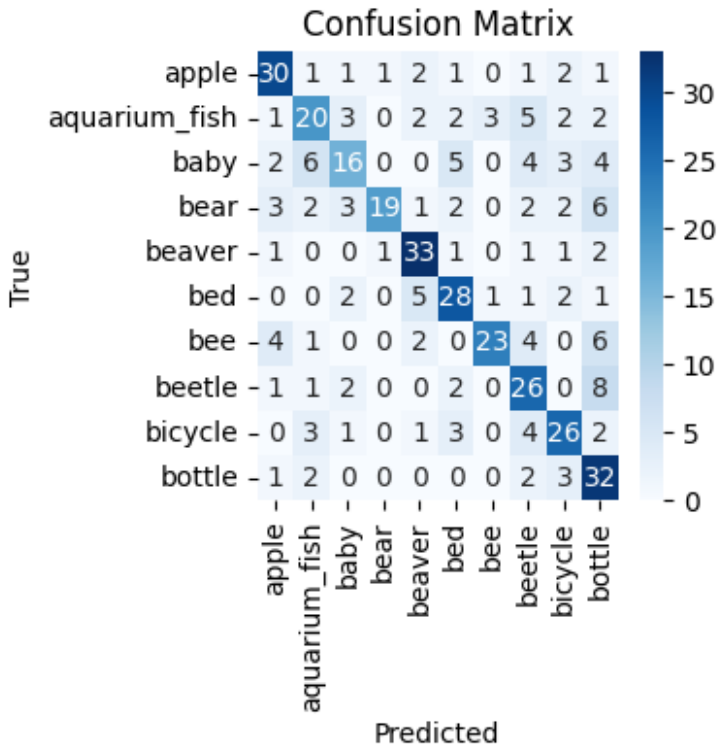
Total training time: 1960.59 seconds Test accuracy: 0.5850 Test loss: 1.4178

Significantly slower, worse accuracy, worse loss

```
# Less convolutional layers
cnn(conv_layers=1)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 125ms/step - accuracy: 0.3498 - loss:
2.0573 - val_accuracy: 0.1325 - val_loss: 2.4081
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 113ms/step - accuracy: 0.6895 - loss:
1.0208 - val_accuracy: 0.1000 - val_loss: 3.4665
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 126ms/step - accuracy: 0.8062 - loss:
0.6414 - val_accuracy: 0.1000 - val_loss: 5.4417
```

```
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 125ms/step - accuracy: 0.9070 - loss:
0.3782 - val_accuracy: 0.1000 - val_loss: 4.4370
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - accuracy: 0.9731 - loss:
0.1894 - val_accuracy: 0.1000 - val_loss: 4.4005
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 132ms/step - accuracy: 0.9799 - loss:
0.1227 - val_accuracy: 0.2000 - val_loss: 3.4627
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 111ms/step - accuracy: 0.9913 - loss:
0.0747 - val_accuracy: 0.3200 - val_loss: 2.6809
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.9945 - loss:
0.0491 - val_accuracy: 0.4775 - val_loss: 1.8619
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 127ms/step - accuracy: 0.9979 - loss:
0.0369 - val_accuracy: 0.4850 - val_loss: 1.8148
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 126ms/step - accuracy: 0.9997 - loss:
0.0235 - val_accuracy: 0.6050 - val_loss: 1.3639
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 114ms/step - accuracy: 1.0000 - loss:
0.0201 - val_accuracy: 0.6325 - val_loss: 1.3077
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 129ms/step - accuracy: 0.9994 - loss:
0.0137 - val_accuracy: 0.6150 - val_loss: 1.3435
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - accuracy: 0.9998 - loss:
0.0119 - val_accuracy: 0.6225 - val_loss: 1.3195
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 110ms/step - accuracy: 1.0000 - loss:
0.0105 - val_accuracy: 0.6075 - val_loss: 1.5427
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.6108 - loss:
1.4454
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step
Total training time: 122.18 seconds
Test accuracy: 0.6325
Test loss: 1.3077
```






```
<keras.src.callbacks.history.History at 0x7a1b92048790>
```

1 convolutional layer

Total training time: 122.18 seconds Test accuracy: 0.6325 Test loss: 1.3077

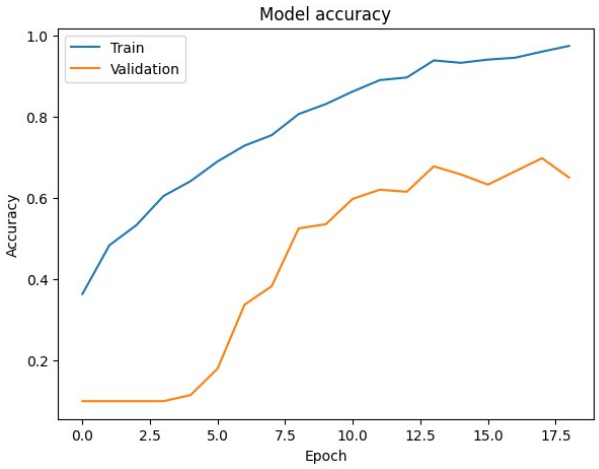
Fast, not as accurate, worse loss, clearly overfits

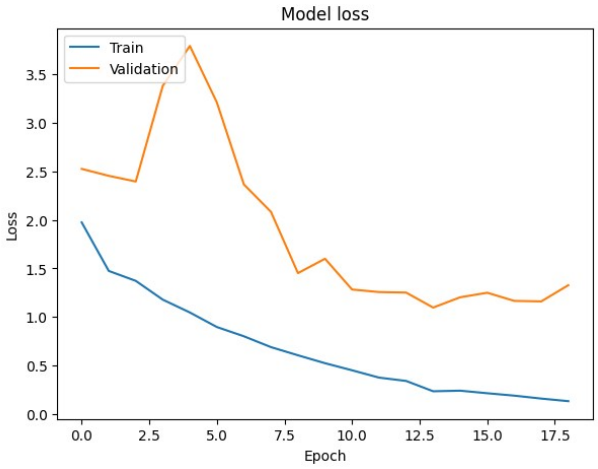
Experiment 3 (activation functions)

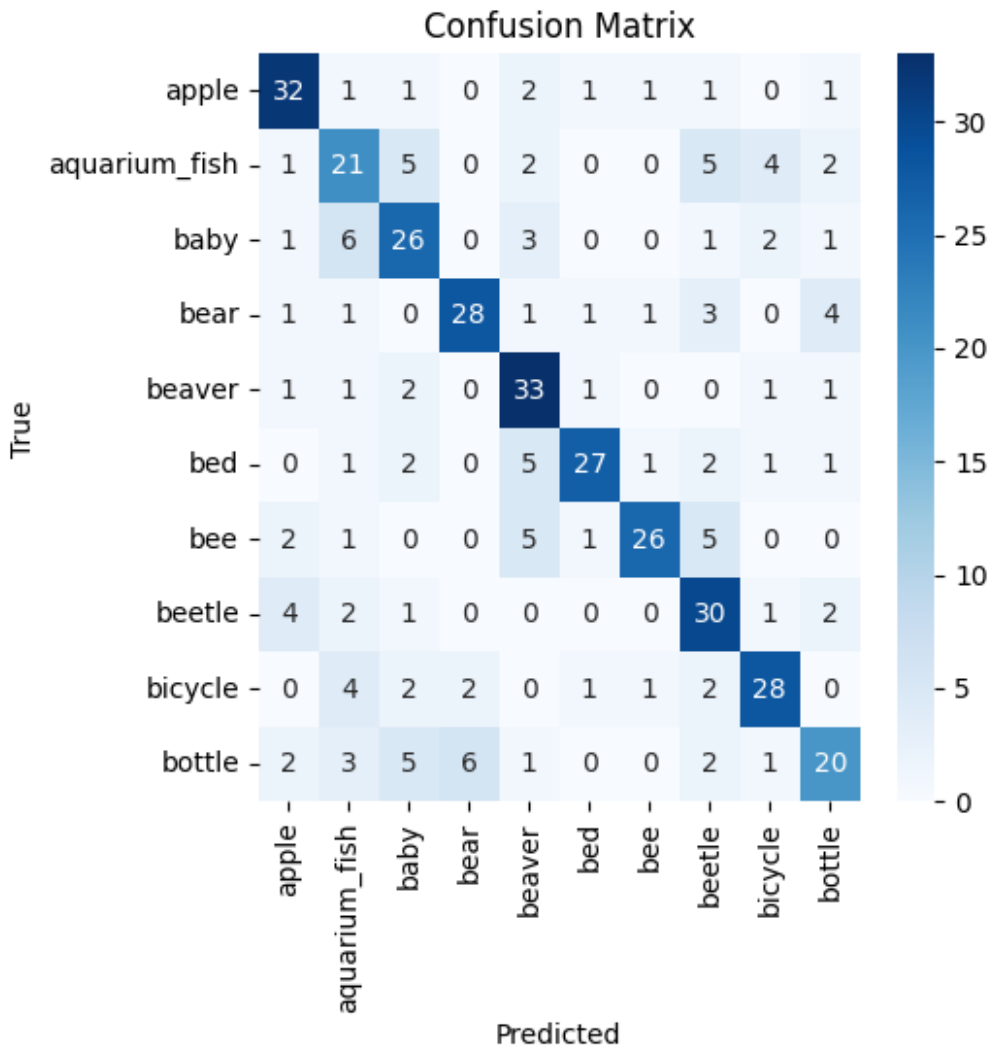
```
# Experiment 3: Different activation functions
cnn(activation='sigmoid')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 28s 327ms/step - accuracy: 0.3014 - loss:
2.2796 - val_accuracy: 0.1000 - val_loss: 2.5265
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 293ms/step - accuracy: 0.4640 - loss:
1.4999 - val_accuracy: 0.1000 - val_loss: 2.4549
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 313ms/step - accuracy: 0.5043 - loss:
1.4666 - val_accuracy: 0.1000 - val_loss: 2.3964
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 293ms/step - accuracy: 0.5996 - loss:
1.1753 - val_accuracy: 0.1000 - val_loss: 3.3840
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 309ms/step - accuracy: 0.6278 - loss:
1.0716 - val_accuracy: 0.1150 - val_loss: 3.7948
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 307ms/step - accuracy: 0.7127 - loss:
0.8586 - val_accuracy: 0.1800 - val_loss: 3.2117
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 313ms/step - accuracy: 0.7402 - loss:
0.7588 - val_accuracy: 0.3375 - val_loss: 2.3655
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 322ms/step - accuracy: 0.7553 - loss:
0.6713 - val_accuracy: 0.3825 - val_loss: 2.0833
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 301ms/step - accuracy: 0.8282 - loss:
0.5686 - val_accuracy: 0.5250 - val_loss: 1.4513
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 323ms/step - accuracy: 0.8388 - loss:
0.4863 - val_accuracy: 0.5350 - val_loss: 1.5996
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 301ms/step - accuracy: 0.8735 - loss:
0.4233 - val_accuracy: 0.5975 - val_loss: 1.2822
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 339ms/step - accuracy: 0.8898 - loss:
0.3684 - val_accuracy: 0.6200 - val_loss: 1.2558
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 297ms/step - accuracy: 0.8986 - loss:
0.3393 - val_accuracy: 0.6150 - val_loss: 1.2505
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 327ms/step - accuracy: 0.9448 - loss:
0.2175 - val_accuracy: 0.6775 - val_loss: 1.0958
```

```
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 296ms/step - accuracy: 0.9458 - loss:
0.2054 - val_accuracy: 0.6575 - val_loss: 1.2020
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 306ms/step - accuracy: 0.9444 - loss:
0.2095 - val_accuracy: 0.6325 - val_loss: 1.2489
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 297ms/step - accuracy: 0.9523 - loss:
0.1743 - val_accuracy: 0.6650 - val_loss: 1.1648
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 299ms/step - accuracy: 0.9604 - loss:
0.1620 - val_accuracy: 0.6975 - val_loss: 1.1595
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 315ms/step - accuracy: 0.9795 - loss:
0.1156 - val_accuracy: 0.6500 - val_loss: 1.3267
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step - accuracy: 0.6916 - loss:
1.1522
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 90ms/step
Total training time: 410.90 seconds
Test accuracy: 0.6775
Test loss: 1.0958
```







Experiment 3, Sigmoid activation function

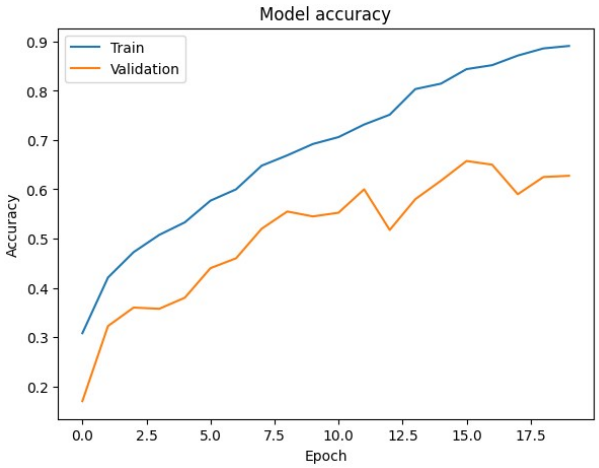
Total training time: 410.90 seconds Test accuracy: 0.6775 Test loss: 1.0958

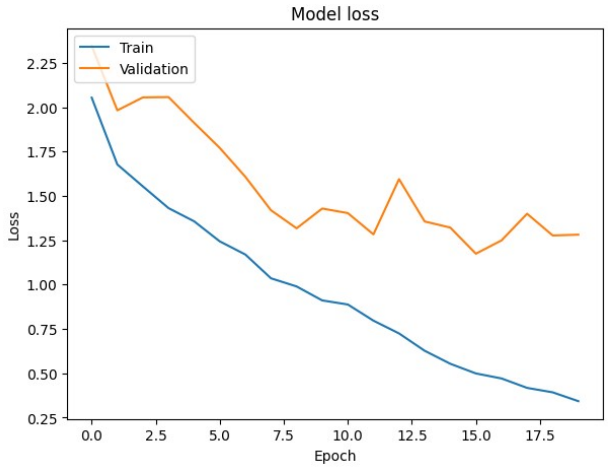
Slower, worse accuracy, worse loss

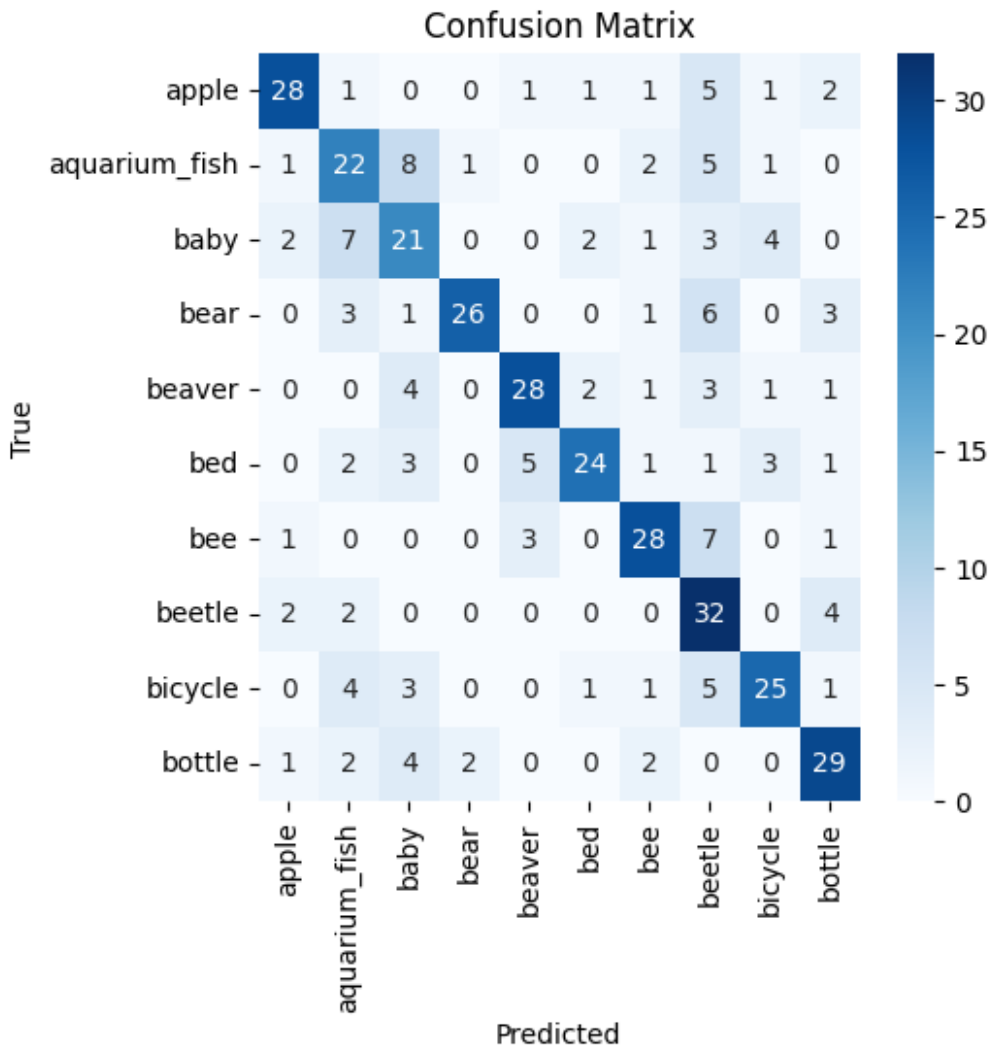
```
cnn(activation='tanh')

/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 314ms/step - accuracy: 0.2569 - loss:
```

```
2.3211 - val_accuracy: 0.1700 - val_loss: 2.3452
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 321ms/step - accuracy: 0.4089 - loss:
1.6978 - val_accuracy: 0.3225 - val_loss: 1.9821
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 294ms/step - accuracy: 0.4781 - loss:
1.5296 - val_accuracy: 0.3600 - val_loss: 2.0552
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 312ms/step - accuracy: 0.5259 - loss:
1.4009 - val_accuracy: 0.3575 - val_loss: 2.0569
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 321ms/step - accuracy: 0.5446 - loss:
1.3069 - val_accuracy: 0.3800 - val_loss: 1.9116
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 302ms/step - accuracy: 0.5787 - loss:
1.2148 - val_accuracy: 0.4400 - val_loss: 1.7704
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 320ms/step - accuracy: 0.5966 - loss:
1.1504 - val_accuracy: 0.4600 - val_loss: 1.6075
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 302ms/step - accuracy: 0.6682 - loss:
0.9808 - val_accuracy: 0.5200 - val_loss: 1.4190
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 327ms/step - accuracy: 0.6909 - loss:
0.9384 - val_accuracy: 0.5550 - val_loss: 1.3173
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 304ms/step - accuracy: 0.7096 - loss:
0.8735 - val_accuracy: 0.5450 - val_loss: 1.4288
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 328ms/step - accuracy: 0.7112 - loss:
0.8630 - val_accuracy: 0.5525 - val_loss: 1.4031
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 303ms/step - accuracy: 0.7378 - loss:
0.7922 - val_accuracy: 0.6000 - val_loss: 1.2832
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 321ms/step - accuracy: 0.7519 - loss:
0.7408 - val_accuracy: 0.5175 - val_loss: 1.5945
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 305ms/step - accuracy: 0.8034 - loss:
0.6230 - val_accuracy: 0.5800 - val_loss: 1.3560
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 318ms/step - accuracy: 0.8080 - loss:
0.5545 - val_accuracy: 0.6175 - val_loss: 1.3214
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 304ms/step - accuracy: 0.8601 - loss:
0.4747 - val_accuracy: 0.6575 - val_loss: 1.1742
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 331ms/step - accuracy: 0.8531 - loss:
0.4602 - val_accuracy: 0.6500 - val_loss: 1.2491
```

```
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 329ms/step - accuracy: 0.8792 - loss:
0.4096 - val_accuracy: 0.5900 - val_loss: 1.3994
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 286ms/step - accuracy: 0.8923 - loss:
0.3913 - val_accuracy: 0.6250 - val_loss: 1.2768
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 297ms/step - accuracy: 0.9006 - loss:
0.3264 - val_accuracy: 0.6275 - val_loss: 1.2814
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step - accuracy: 0.6385 - loss:
1.2684
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 82ms/step
Total training time: 474.09 seconds
Test accuracy: 0.6575
Test loss: 1.1742
```







Tanh activation function

Total training time: 474.09 seconds Test accuracy: 0.6575 Test loss: 1.1742

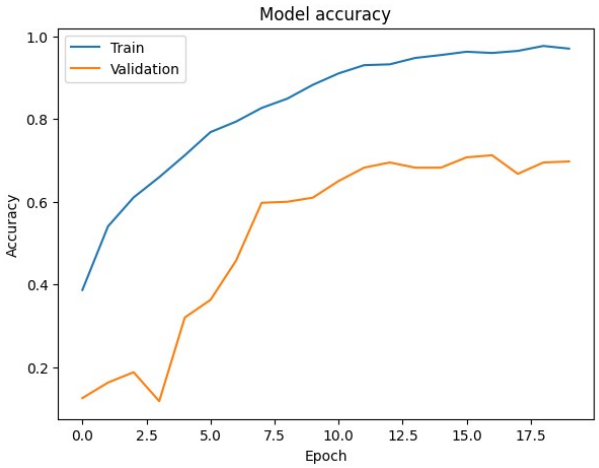
Slower, worse accuracy, worse loss

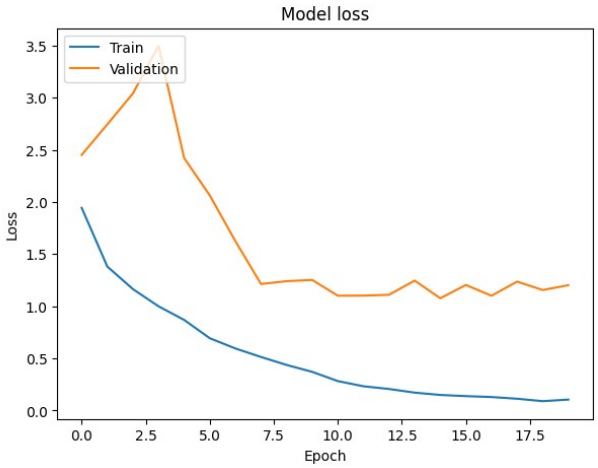
```
cnn(activation='leaky_relu')
```

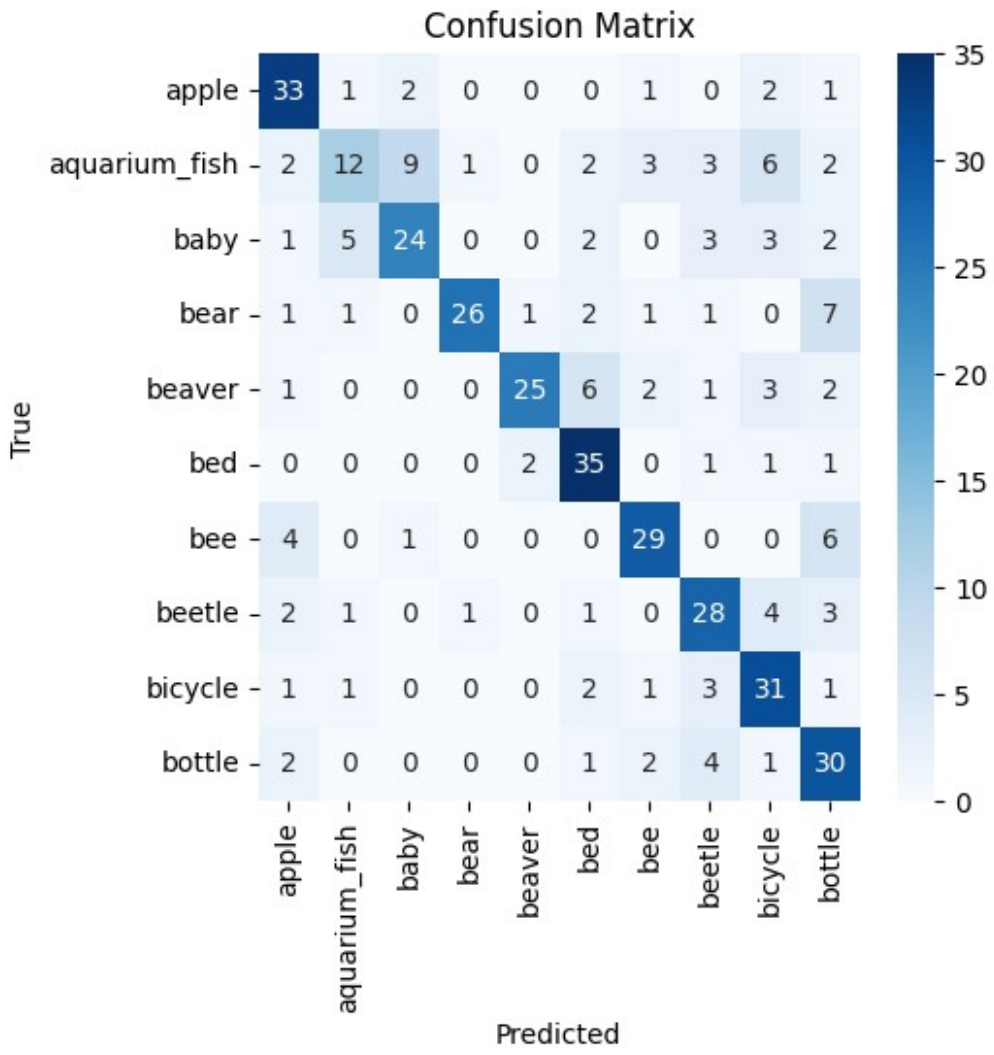
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 26s 300ms/step - accuracy: 0.3149 - loss:
```

```
2.2231 - val_accuracy: 0.1250 - val_loss: 2.4518
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 294ms/step - accuracy: 0.5345 - loss:
1.3637 - val_accuracy: 0.1625 - val_loss: 2.7445
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 289ms/step - accuracy: 0.5922 - loss:
1.2211 - val_accuracy: 0.1875 - val_loss: 3.0413
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 315ms/step - accuracy: 0.6788 - loss:
0.9777 - val_accuracy: 0.1175 - val_loss: 3.4963
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 291ms/step - accuracy: 0.7180 - loss:
0.8600 - val_accuracy: 0.3200 - val_loss: 2.4206
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 309ms/step - accuracy: 0.7805 - loss:
0.6751 - val_accuracy: 0.3625 - val_loss: 2.0631
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 320ms/step - accuracy: 0.7961 - loss:
0.5900 - val_accuracy: 0.4575 - val_loss: 1.6245
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 291ms/step - accuracy: 0.8389 - loss:
0.4781 - val_accuracy: 0.5975 - val_loss: 1.2133
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 336ms/step - accuracy: 0.8604 - loss:
0.4204 - val_accuracy: 0.6000 - val_loss: 1.2405
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 294ms/step - accuracy: 0.8847 - loss:
0.3604 - val_accuracy: 0.6100 - val_loss: 1.2522
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 302ms/step - accuracy: 0.9182 - loss:
0.2781 - val_accuracy: 0.6500 - val_loss: 1.1009
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 292ms/step - accuracy: 0.9353 - loss:
0.2154 - val_accuracy: 0.6825 - val_loss: 1.1019
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 315ms/step - accuracy: 0.9376 - loss:
0.1964 - val_accuracy: 0.6950 - val_loss: 1.1099
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 317ms/step - accuracy: 0.9463 - loss:
0.1700 - val_accuracy: 0.6825 - val_loss: 1.2460
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 297ms/step - accuracy: 0.9656 - loss:
0.1380 - val_accuracy: 0.6825 - val_loss: 1.0764
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 324ms/step - accuracy: 0.9692 - loss:
0.1255 - val_accuracy: 0.7075 - val_loss: 1.2036
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 306ms/step - accuracy: 0.9524 - loss:
0.1410 - val_accuracy: 0.7125 - val_loss: 1.1008
```

```
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 321ms/step - accuracy: 0.9706 - loss:
0.0960 - val_accuracy: 0.6675 - val_loss: 1.2365
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 295ms/step - accuracy: 0.9805 - loss:
0.0854 - val_accuracy: 0.6950 - val_loss: 1.1554
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 293ms/step - accuracy: 0.9756 - loss:
0.0986 - val_accuracy: 0.6975 - val_loss: 1.2018
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 64ms/step - accuracy: 0.6542 - loss:
1.2732
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 120ms/step
Total training time: 433.51 seconds
Test accuracy: 0.6825
Test loss: 1.0764
```







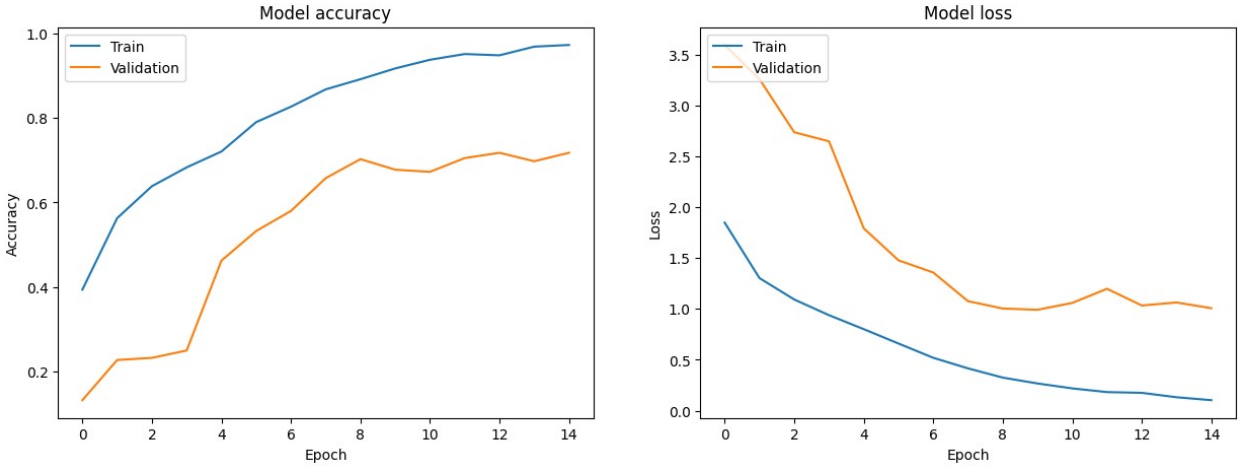
Leaky relu

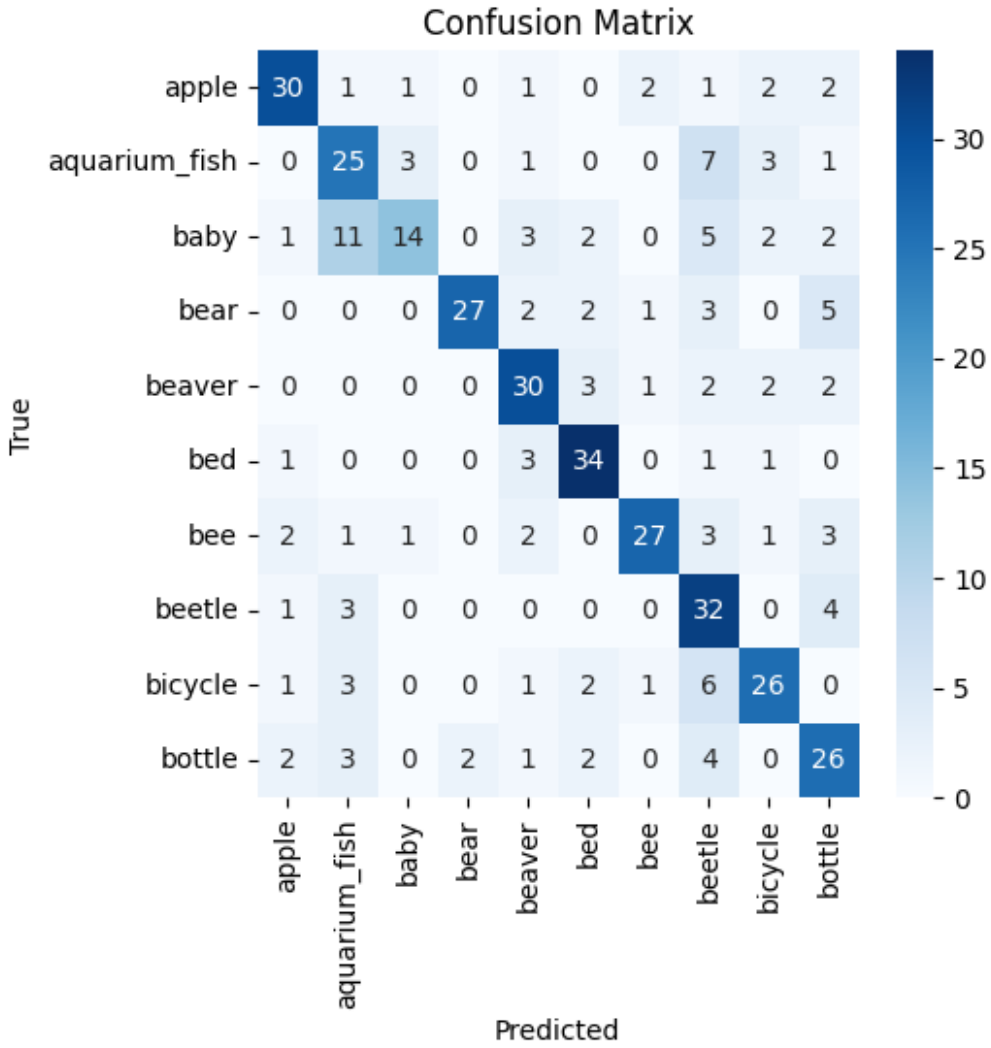
Total training time: 433.51 seconds Test accuracy: 0.6825 Test loss: 1.0764

Slower, otherwise pretty much the same as baseline

```
cnn(activation='elu')

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 44s 374ms/step - accuracy: 0.3234 - loss:
2.1798 - val_accuracy: 0.1325 - val_loss: 3.5957
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 319ms/step - accuracy: 0.5536 - loss:
1.3162 - val_accuracy: 0.2275 - val_loss: 3.2595
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 329ms/step - accuracy: 0.6432 - loss:
1.0822 - val_accuracy: 0.2325 - val_loss: 2.7375
Epoch 4/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 43s 369ms/step - accuracy: 0.6768 - loss:
0.9511 - val_accuracy: 0.2500 - val_loss: 2.6483
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 316ms/step - accuracy: 0.7248 - loss:
0.7916 - val_accuracy: 0.4625 - val_loss: 1.7930
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 354ms/step - accuracy: 0.7884 - loss:
0.6632 - val_accuracy: 0.5325 - val_loss: 1.4779
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 325ms/step - accuracy: 0.8309 - loss:
0.5117 - val_accuracy: 0.5800 - val_loss: 1.3584
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 314ms/step - accuracy: 0.8741 - loss:
0.4094 - val_accuracy: 0.6575 - val_loss: 1.0764
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 337ms/step - accuracy: 0.9025 - loss:
0.3108 - val_accuracy: 0.7025 - val_loss: 1.0044
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 315ms/step - accuracy: 0.9307 - loss:
0.2488 - val_accuracy: 0.6775 - val_loss: 0.9920
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 298ms/step - accuracy: 0.9380 - loss:
0.2148 - val_accuracy: 0.6725 - val_loss: 1.0595
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 26s 418ms/step - accuracy: 0.9551 - loss:
0.1710 - val_accuracy: 0.7050 - val_loss: 1.1983
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 36s 333ms/step - accuracy: 0.9507 - loss:
0.1675 - val_accuracy: 0.7175 - val_loss: 1.0339
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 325ms/step - accuracy: 0.9666 - loss:
0.1265 - val_accuracy: 0.6975 - val_loss: 1.0636
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 351ms/step - accuracy: 0.9661 - loss:
0.1038 - val_accuracy: 0.7175 - val_loss: 1.0077
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step - accuracy: 0.6591 - loss:
1.0594
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 90ms/step
Total training time: 493.11 seconds
Test accuracy: 0.6775
Test loss: 0.9920
```





Elu

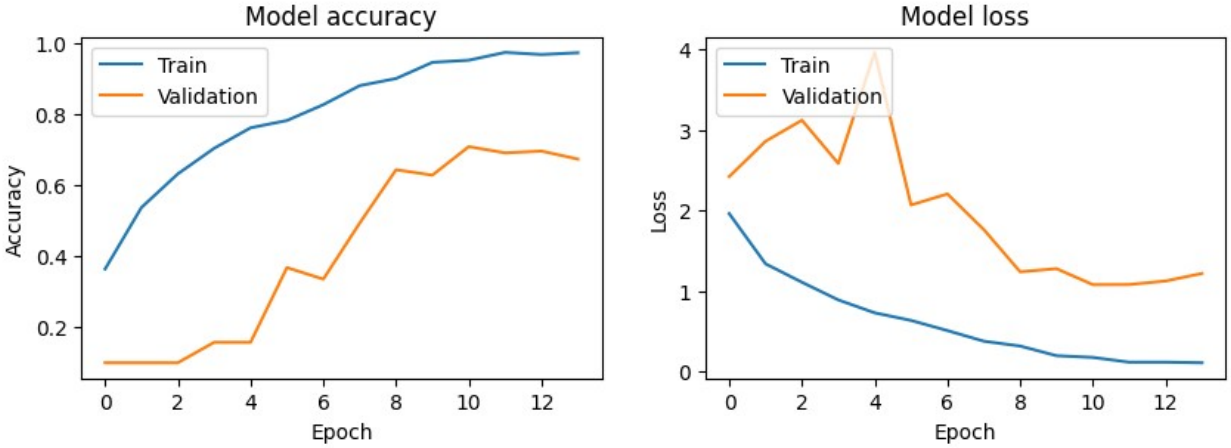
Total training time: 493.11 seconds Test accuracy: 0.6775 Test loss: 0.9920

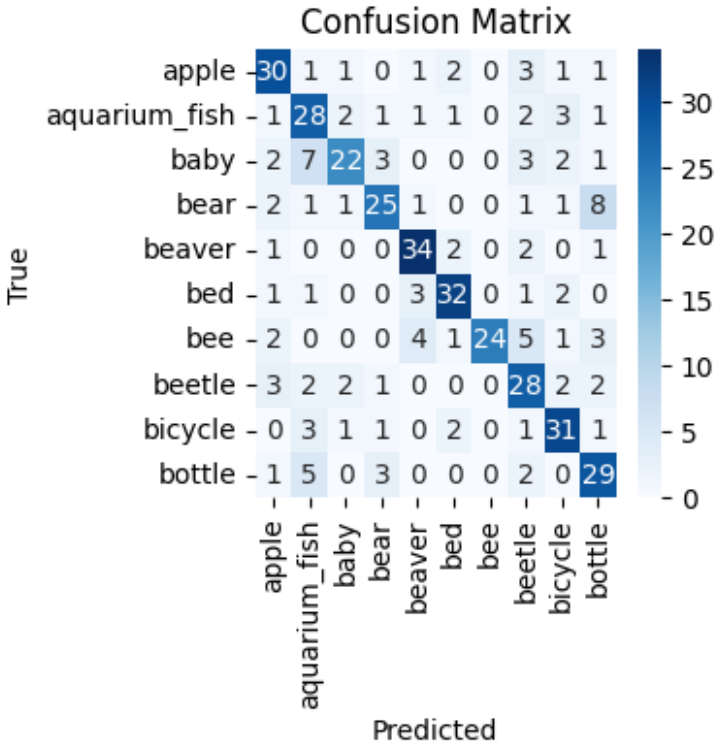
Slower, worse accuracy, better loss

```
cnn(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 33s 391ms/step - accuracy: 0.2891 - loss:
2.2838 - val_accuracy: 0.1000 - val_loss: 2.4177
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 297ms/step - accuracy: 0.5406 - loss:
1.3187 - val_accuracy: 0.1000 - val_loss: 2.8565
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 277ms/step - accuracy: 0.6242 - loss:
1.1366 - val_accuracy: 0.1000 - val_loss: 3.1174
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 286ms/step - accuracy: 0.7022 - loss:
0.8804 - val_accuracy: 0.1575 - val_loss: 2.5813
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 277ms/step - accuracy: 0.7713 - loss:
0.7103 - val_accuracy: 0.1575 - val_loss: 3.9588
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.7998 - loss:
0.5980 - val_accuracy: 0.3675 - val_loss: 2.0660
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 303ms/step - accuracy: 0.8300 - loss:
0.4912 - val_accuracy: 0.3350 - val_loss: 2.2030
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 287ms/step - accuracy: 0.8877 - loss:
0.3586 - val_accuracy: 0.4925 - val_loss: 1.7613
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 291ms/step - accuracy: 0.9164 - loss:
0.2933 - val_accuracy: 0.6425 - val_loss: 1.2372
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 276ms/step - accuracy: 0.9408 - loss:
0.2053 - val_accuracy: 0.6275 - val_loss: 1.2758
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.9577 - loss:
0.1555 - val_accuracy: 0.7075 - val_loss: 1.0776
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 277ms/step - accuracy: 0.9764 - loss:
0.1145 - val_accuracy: 0.6900 - val_loss: 1.0791
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 276ms/step - accuracy: 0.9696 - loss:
```

```
0.1060 - val_accuracy: 0.6950 - val_loss: 1.1230
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 294ms/step - accuracy: 0.9762 - loss:
0.0969 - val_accuracy: 0.6725 - val_loss: 1.2166
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 62ms/step - accuracy: 0.7034 - loss:
1.2018
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 81ms/step
Total training time: 291.47 seconds
Test accuracy: 0.7075
Test loss: 1.0776
```





```
<keras.src.callbacks.history.History at 0x7a1b9215b0d0>
```

Swish

Total training time: 291.47 seconds Test accuracy: 0.7075 Test loss: 1.0776

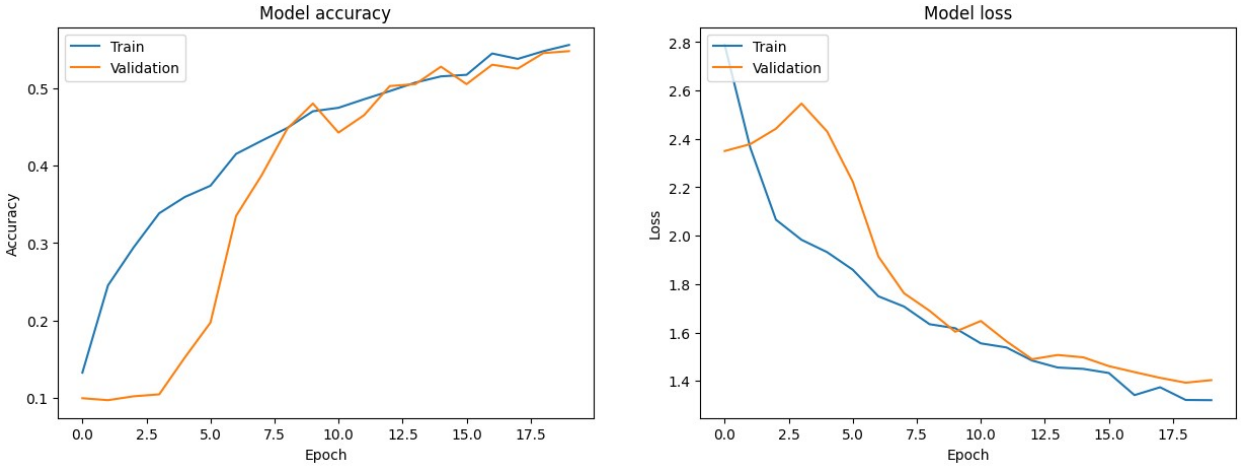
Better accuracy, about the same loss, but signs of overfitting

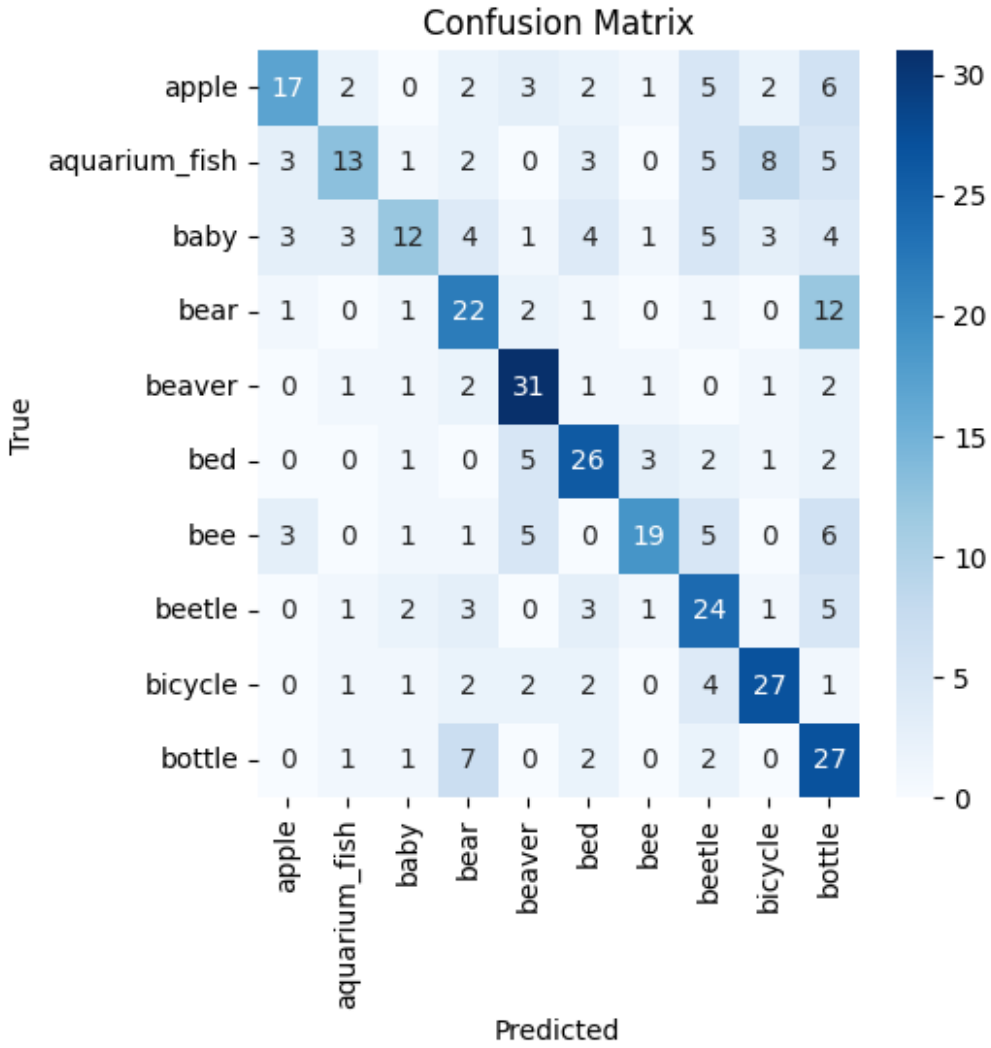
Experiment 4 (optimizers and learning rate)

```
# Experiment 4: Different optimizers and learning rates
model_4 = cnn(optimizer_name='sgd')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 31s 443ms/step - accuracy: 0.1021 - loss:
2.9325 - val_accuracy: 0.1000 - val_loss: 2.3500
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 33s 316ms/step - accuracy: 0.2306 - loss:
2.4019 - val_accuracy: 0.0975 - val_loss: 2.3785
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 298ms/step - accuracy: 0.2865 - loss:
2.1126 - val_accuracy: 0.1025 - val_loss: 2.4422
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 336ms/step - accuracy: 0.3269 - loss:
2.0019 - val_accuracy: 0.1050 - val_loss: 2.5461
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 297ms/step - accuracy: 0.3466 - loss:
1.9743 - val_accuracy: 0.1525 - val_loss: 2.4305
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 324ms/step - accuracy: 0.3584 - loss:
1.8895 - val_accuracy: 0.1975 - val_loss: 2.2233
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 298ms/step - accuracy: 0.4194 - loss:
1.7634 - val_accuracy: 0.3350 - val_loss: 1.9141
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 297ms/step - accuracy: 0.4540 - loss:
1.6580 - val_accuracy: 0.3875 - val_loss: 1.7624
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 319ms/step - accuracy: 0.4349 - loss:
1.6867 - val_accuracy: 0.4475 - val_loss: 1.6897
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 301ms/step - accuracy: 0.4745 - loss:
```

```
1.6062 - val_accuracy: 0.4800 - val_loss: 1.6032
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 313ms/step - accuracy: 0.4487 - loss:
1.6153 - val_accuracy: 0.4425 - val_loss: 1.6477
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 295ms/step - accuracy: 0.4846 - loss:
1.5637 - val_accuracy: 0.4650 - val_loss: 1.5638
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 324ms/step - accuracy: 0.4917 - loss:
1.4741 - val_accuracy: 0.5025 - val_loss: 1.4896
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 298ms/step - accuracy: 0.4975 - loss:
1.4690 - val_accuracy: 0.5050 - val_loss: 1.5072
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 310ms/step - accuracy: 0.5070 - loss:
1.4857 - val_accuracy: 0.5275 - val_loss: 1.4975
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 323ms/step - accuracy: 0.5391 - loss:
1.4107 - val_accuracy: 0.5050 - val_loss: 1.4611
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 297ms/step - accuracy: 0.5505 - loss:
1.3596 - val_accuracy: 0.5300 - val_loss: 1.4364
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 345ms/step - accuracy: 0.5403 - loss:
1.3703 - val_accuracy: 0.5250 - val_loss: 1.4127
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 309ms/step - accuracy: 0.5449 - loss:
1.3160 - val_accuracy: 0.5450 - val_loss: 1.3927
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 305ms/step - accuracy: 0.5716 - loss:
1.2621 - val_accuracy: 0.5475 - val_loss: 1.4033
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - accuracy: 0.4682 - loss:
1.6714
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 94ms/step
Total training time: 450.70 seconds
Test accuracy: 0.5450
Test loss: 1.3927
```





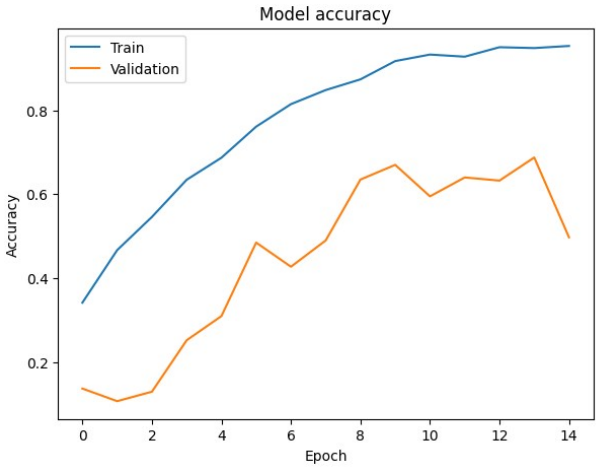
Total training time: 450.70 seconds Test accuracy: 0.5450 Test loss: 1.3927

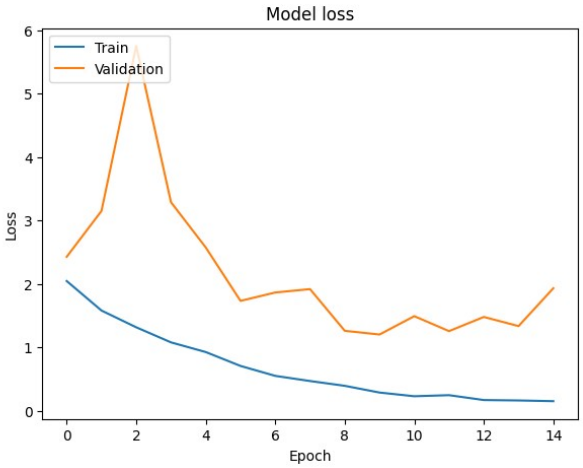
Worse accuracy and loss

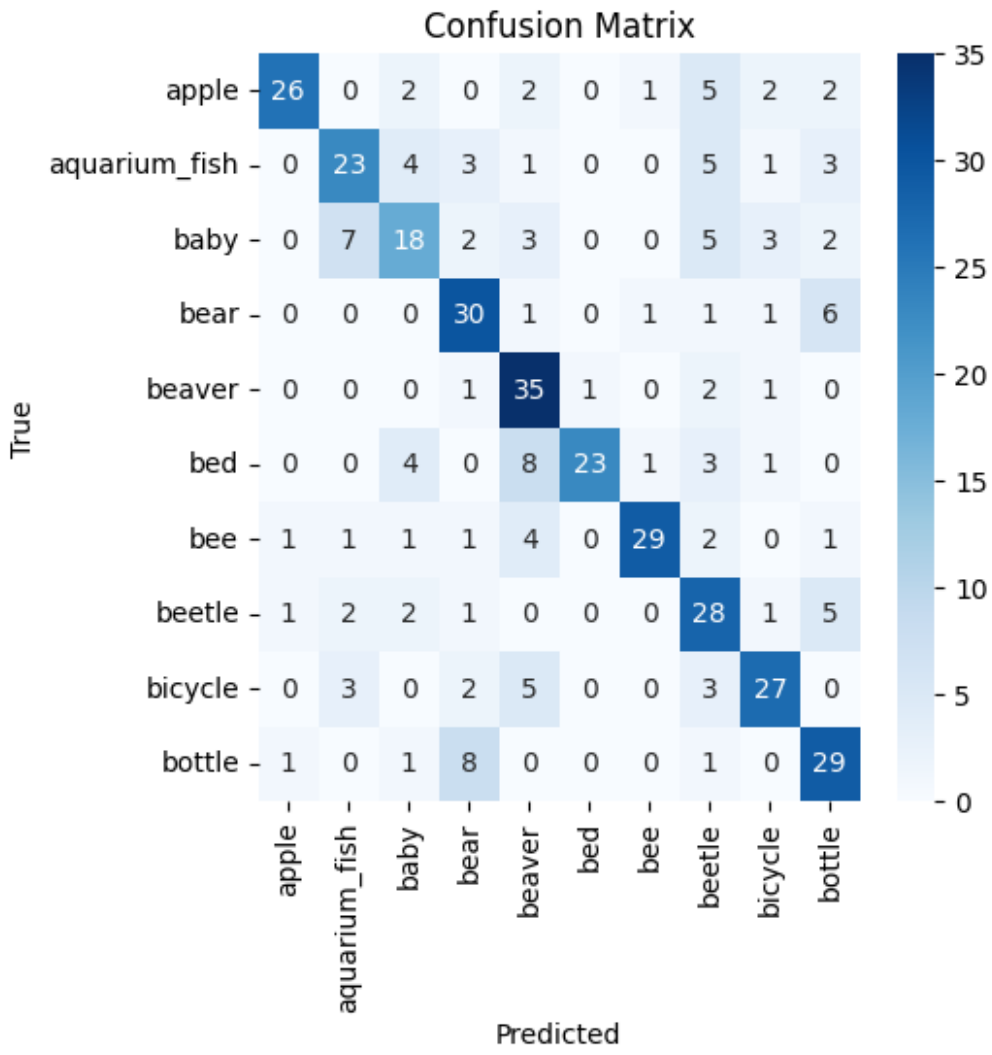
```
model_4b = cnn(optimizer_name='rmsprop')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 300ms/step - accuracy: 0.2864 - loss:
2.2933 - val_accuracy: 0.1375 - val_loss: 2.4264
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 304ms/step - accuracy: 0.4685 - loss:
1.5975 - val_accuracy: 0.1075 - val_loss: 3.1493
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 311ms/step - accuracy: 0.5431 - loss:
1.3070 - val_accuracy: 0.1300 - val_loss: 5.7512
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 298ms/step - accuracy: 0.6385 - loss:
1.0609 - val_accuracy: 0.2525 - val_loss: 3.2864
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 324ms/step - accuracy: 0.7057 - loss:
0.8824 - val_accuracy: 0.3100 - val_loss: 2.5737
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 292ms/step - accuracy: 0.7739 - loss:
0.6769 - val_accuracy: 0.4850 - val_loss: 1.7309
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 322ms/step - accuracy: 0.8256 - loss:
0.5255 - val_accuracy: 0.4275 - val_loss: 1.8630
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 296ms/step - accuracy: 0.8547 - loss:
0.4571 - val_accuracy: 0.4900 - val_loss: 1.9168
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 310ms/step - accuracy: 0.8774 - loss:
0.3777 - val_accuracy: 0.6350 - val_loss: 1.2588
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 291ms/step - accuracy: 0.9315 - loss:
0.2548 - val_accuracy: 0.6700 - val_loss: 1.2006
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 327ms/step - accuracy: 0.9319 - loss:
0.2268 - val_accuracy: 0.5950 - val_loss: 1.4898
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 292ms/step - accuracy: 0.9273 - loss:
0.2225 - val_accuracy: 0.6400 - val_loss: 1.2539
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 314ms/step - accuracy: 0.9559 - loss:
0.1443 - val_accuracy: 0.6325 - val_loss: 1.4774
Epoch 14/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 295ms/step - accuracy: 0.9581 - loss:
0.1372 - val_accuracy: 0.6875 - val_loss: 1.3333
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 314ms/step - accuracy: 0.9680 - loss:
0.1175 - val_accuracy: 0.4975 - val_loss: 1.9302
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step - accuracy: 0.6442 - loss:
1.3160
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 77ms/step
Total training time: 303.18 seconds
Test accuracy: 0.6700
Test loss: 1.2006
```







Total training time: 303.18 seconds Test accuracy: 0.6700 Test loss: 1.2006

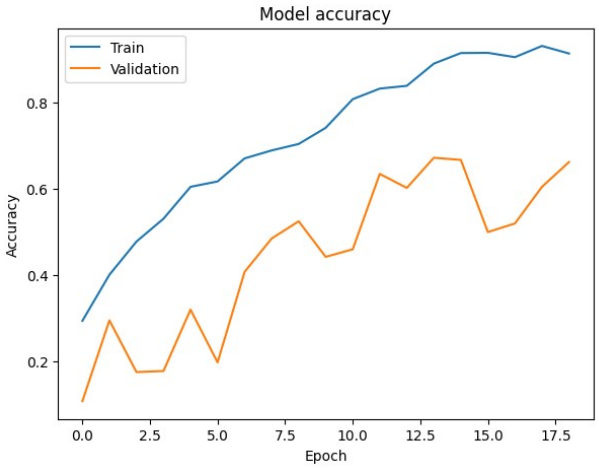
Worse accuracy, worse loss

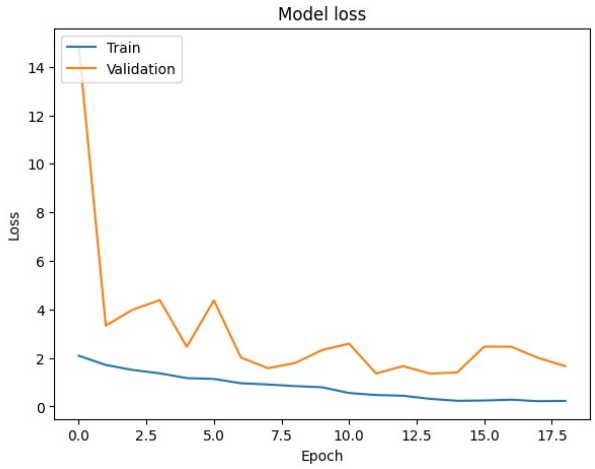
```
model_4c = cnn(optimizer_name='adam', learning_rate=0.01)
```

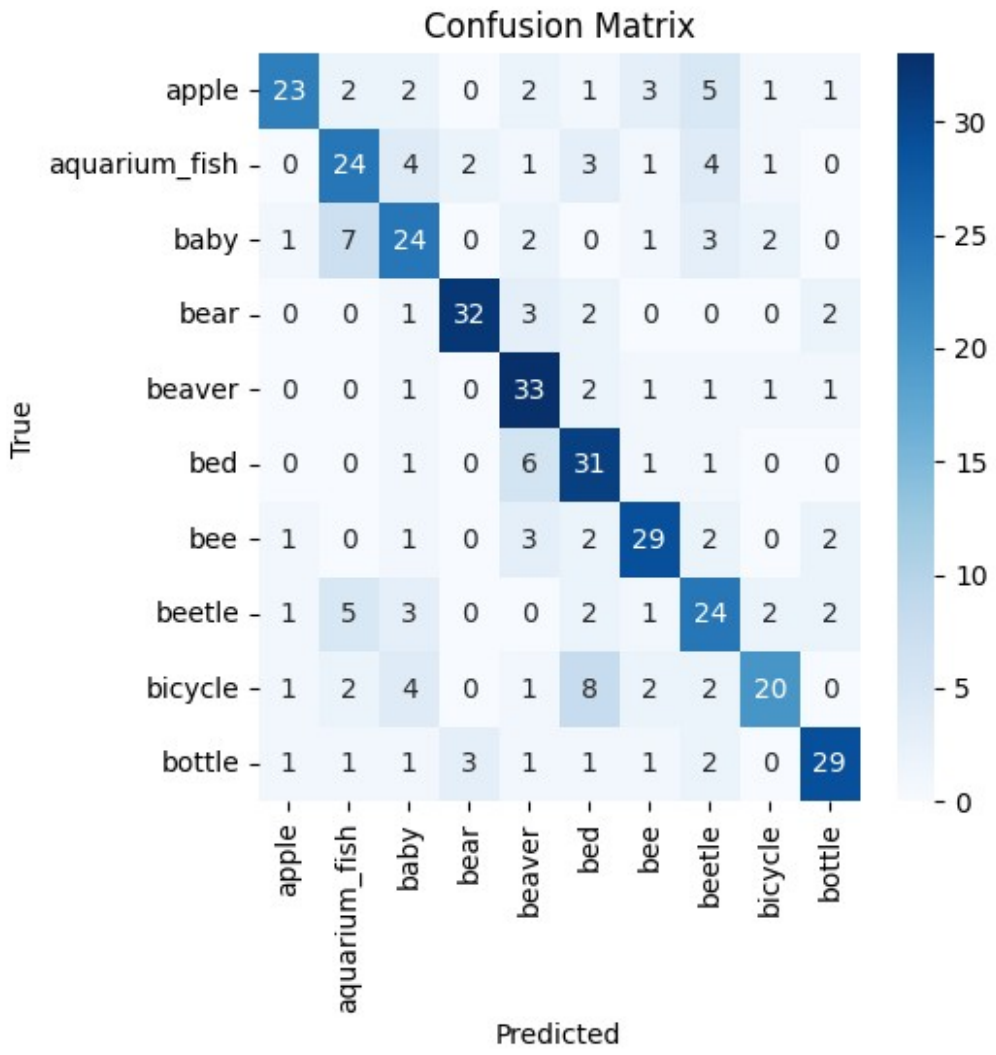
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 304ms/step - accuracy: 0.2621 - loss:
2.3467 - val_accuracy: 0.1075 - val_loss: 14.8562
Epoch 2/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 316ms/step - accuracy: 0.3788 - loss:
1.7734 - val_accuracy: 0.2950 - val_loss: 3.3319
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 299ms/step - accuracy: 0.4745 - loss:
1.5091 - val_accuracy: 0.1750 - val_loss: 3.9918
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 326ms/step - accuracy: 0.5428 - loss:
1.3738 - val_accuracy: 0.1775 - val_loss: 4.3823
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 309ms/step - accuracy: 0.6156 - loss:
1.1492 - val_accuracy: 0.3200 - val_loss: 2.4558
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 324ms/step - accuracy: 0.6143 - loss:
1.1266 - val_accuracy: 0.1975 - val_loss: 4.3734
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 305ms/step - accuracy: 0.6765 - loss:
0.9323 - val_accuracy: 0.4075 - val_loss: 2.0173
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 352ms/step - accuracy: 0.6761 - loss:
0.9117 - val_accuracy: 0.4850 - val_loss: 1.5766
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 305ms/step - accuracy: 0.7290 - loss:
0.7864 - val_accuracy: 0.5250 - val_loss: 1.7891
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 325ms/step - accuracy: 0.7310 - loss:
0.8431 - val_accuracy: 0.4425 - val_loss: 2.3230
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 294ms/step - accuracy: 0.8195 - loss:
0.5223 - val_accuracy: 0.4600 - val_loss: 2.5887
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 318ms/step - accuracy: 0.8360 - loss:
0.4501 - val_accuracy: 0.6350 - val_loss: 1.3617
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 302ms/step - accuracy: 0.8322 - loss:
0.4463 - val_accuracy: 0.6025 - val_loss: 1.6628
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 314ms/step - accuracy: 0.9002 - loss:
0.3081 - val_accuracy: 0.6725 - val_loss: 1.3522
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 296ms/step - accuracy: 0.9357 - loss:
0.1980 - val_accuracy: 0.6675 - val_loss: 1.4002
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 322ms/step - accuracy: 0.9236 - loss:
0.2170 - val_accuracy: 0.5000 - val_loss: 2.4657
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 363ms/step - accuracy: 0.9125 - loss:
0.2587 - val_accuracy: 0.5200 - val_loss: 2.4612
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 325ms/step - accuracy: 0.9338 - loss:
```

```
0.2306 - val_accuracy: 0.6050 - val_loss: 1.9987
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 299ms/step - accuracy: 0.9152 - loss:
0.2278 - val_accuracy: 0.6625 - val_loss: 1.6616
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 88ms/step - accuracy: 0.6534 - loss:
1.5151
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 127ms/step
Total training time: 455.27 seconds
Test accuracy: 0.6725
Test loss: 1.3522
```







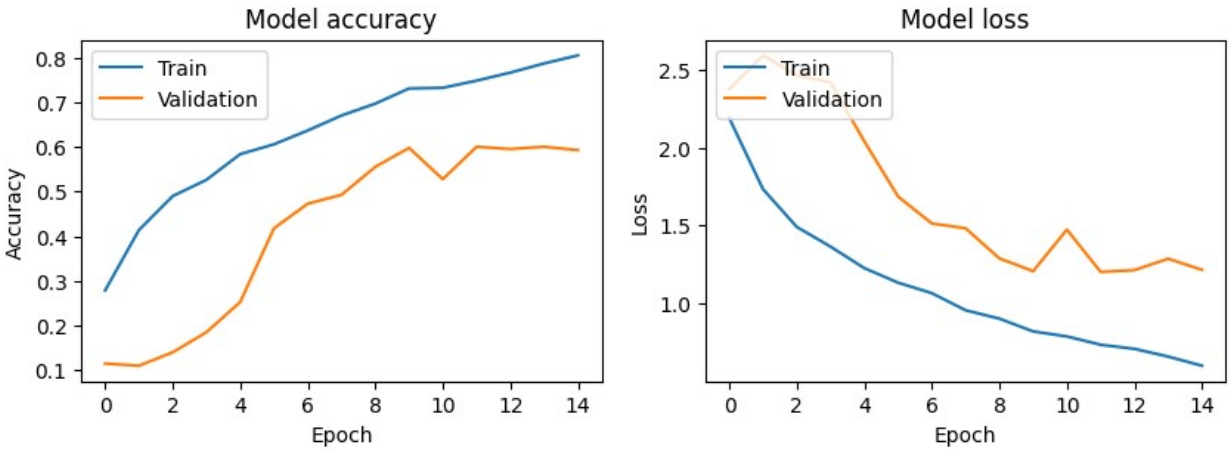
Total training time: 455.27 seconds Test accuracy: 0.6725 Test loss: 1.3522

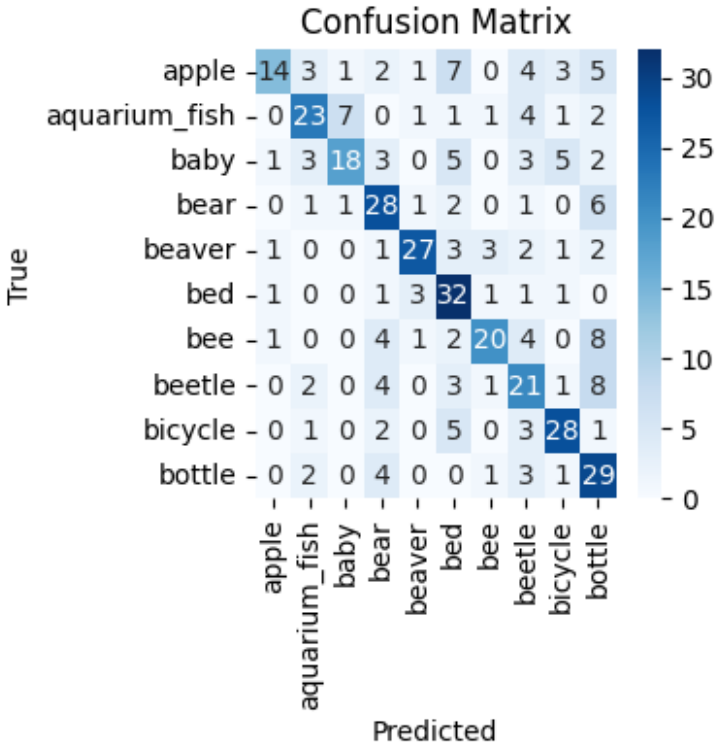
Slower, about the same accuracy, worse loss

```
model_4d = cnn(optimizer_name='sgd', learning_rate=0.01)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 278ms/step - accuracy: 0.2119 - loss:
2.4571 - val_accuracy: 0.1150 - val_loss: 2.3721
Epoch 2/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 263ms/step - accuracy: 0.3958 - loss:
1.8000 - val_accuracy: 0.1100 - val_loss: 2.5911
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 252ms/step - accuracy: 0.4808 - loss:
1.5190 - val_accuracy: 0.1400 - val_loss: 2.4616
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 261ms/step - accuracy: 0.5266 - loss:
1.3564 - val_accuracy: 0.1850 - val_loss: 2.4179
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 258ms/step - accuracy: 0.5682 - loss:
1.2723 - val_accuracy: 0.2525 - val_loss: 2.0402
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 250ms/step - accuracy: 0.6032 - loss:
1.1283 - val_accuracy: 0.4175 - val_loss: 1.6857
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 278ms/step - accuracy: 0.6481 - loss:
1.0271 - val_accuracy: 0.4725 - val_loss: 1.5131
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 264ms/step - accuracy: 0.6810 - loss:
0.9333 - val_accuracy: 0.4925 - val_loss: 1.4818
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 254ms/step - accuracy: 0.6954 - loss:
0.9030 - val_accuracy: 0.5550 - val_loss: 1.2888
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 273ms/step - accuracy: 0.7578 - loss:
0.7871 - val_accuracy: 0.5975 - val_loss: 1.2069
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 262ms/step - accuracy: 0.7576 - loss:
0.7403 - val_accuracy: 0.5275 - val_loss: 1.4735
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 262ms/step - accuracy: 0.7630 - loss:
0.7235 - val_accuracy: 0.6000 - val_loss: 1.2019
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 264ms/step - accuracy: 0.7691 - loss:
0.7099 - val_accuracy: 0.5950 - val_loss: 1.2138
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 263ms/step - accuracy: 0.7880 - loss:
0.6489 - val_accuracy: 0.6000 - val_loss: 1.2866
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 265ms/step - accuracy: 0.8061 - loss:
0.5992 - val_accuracy: 0.5925 - val_loss: 1.2155
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 56ms/step - accuracy: 0.5347 - loss:
1.3706
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 76ms/step
Total training time: 298.17 seconds
Test accuracy: 0.6000
Test loss: 1.2019
```





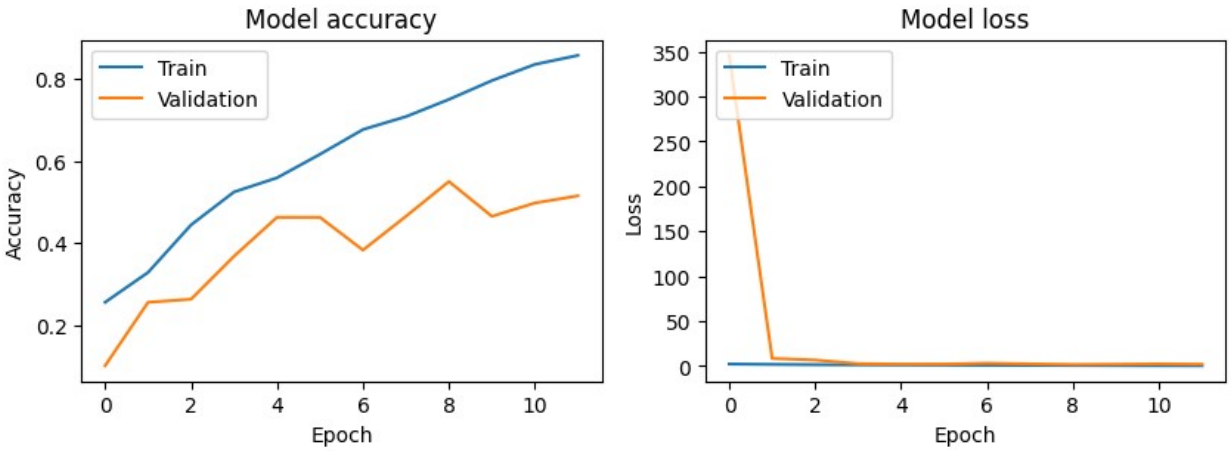
Total training time: 298.17 seconds Test accuracy: 0.6000 Test loss: 1.2019

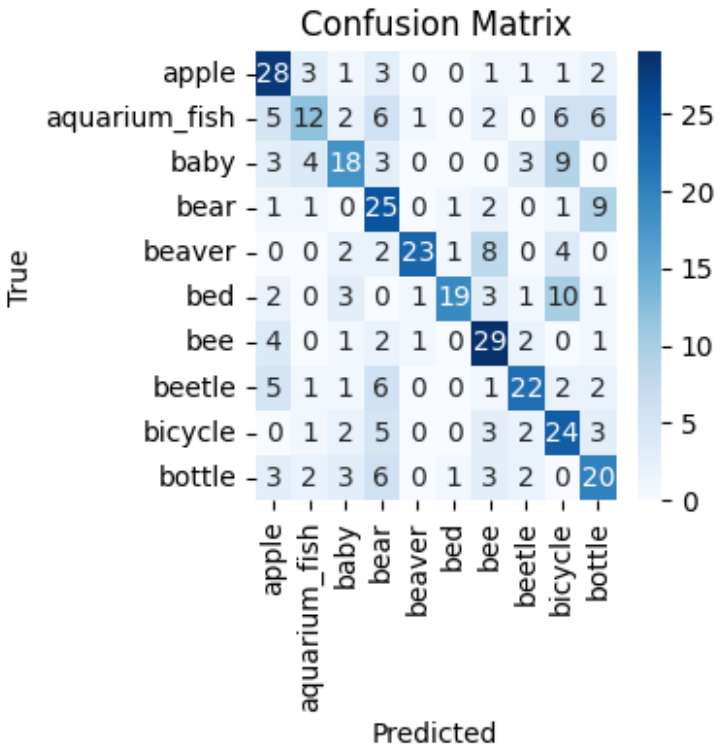
Worse accuracy, worse loss

```
model_4e = cnn(optimizer_name='rmsprop', learning_rate=0.01)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
```

```
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 291ms/step - accuracy: 0.2077 - loss:
2.6979 - val_accuracy: 0.1000 - val_loss: 345.7508
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 314ms/step - accuracy: 0.3124 - loss:
1.9915 - val_accuracy: 0.2550 - val_loss: 8.5585
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 323ms/step - accuracy: 0.4335 - loss:
1.6362 - val_accuracy: 0.2625 - val_loss: 6.7809
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 315ms/step - accuracy: 0.5434 - loss:
1.3453 - val_accuracy: 0.3675 - val_loss: 2.7077
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 320ms/step - accuracy: 0.5638 - loss:
1.2135 - val_accuracy: 0.4625 - val_loss: 2.1475
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 308ms/step - accuracy: 0.6255 - loss:
1.0624 - val_accuracy: 0.4625 - val_loss: 2.1741
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 305ms/step - accuracy: 0.6869 - loss:
0.9080 - val_accuracy: 0.3825 - val_loss: 3.2052
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 309ms/step - accuracy: 0.7297 - loss:
0.7862 - val_accuracy: 0.4650 - val_loss: 2.4013
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 278ms/step - accuracy: 0.7571 - loss:
0.7135 - val_accuracy: 0.5500 - val_loss: 1.7608
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 25s 355ms/step - accuracy: 0.8148 - loss:
0.5483 - val_accuracy: 0.4650 - val_loss: 1.9912
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 322ms/step - accuracy: 0.8467 - loss:
0.4878 - val_accuracy: 0.4975 - val_loss: 2.3786
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 309ms/step - accuracy: 0.8687 - loss:
0.3773 - val_accuracy: 0.5150 - val_loss: 2.0699
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.5487 - loss:
1.8187
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 131ms/step
Total training time: 270.07 seconds
Test accuracy: 0.5500
Test loss: 1.7608
```





Total training time: 270.07 seconds Test accuracy: 0.5500 Test loss: 1.7608

Worse accuracy and loss

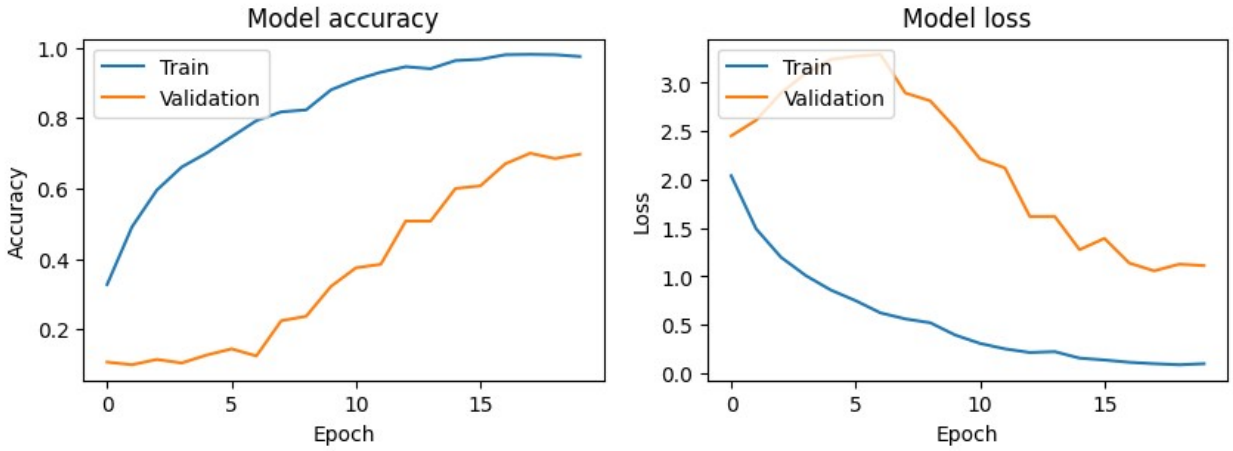
Experiment 5 (batch size)

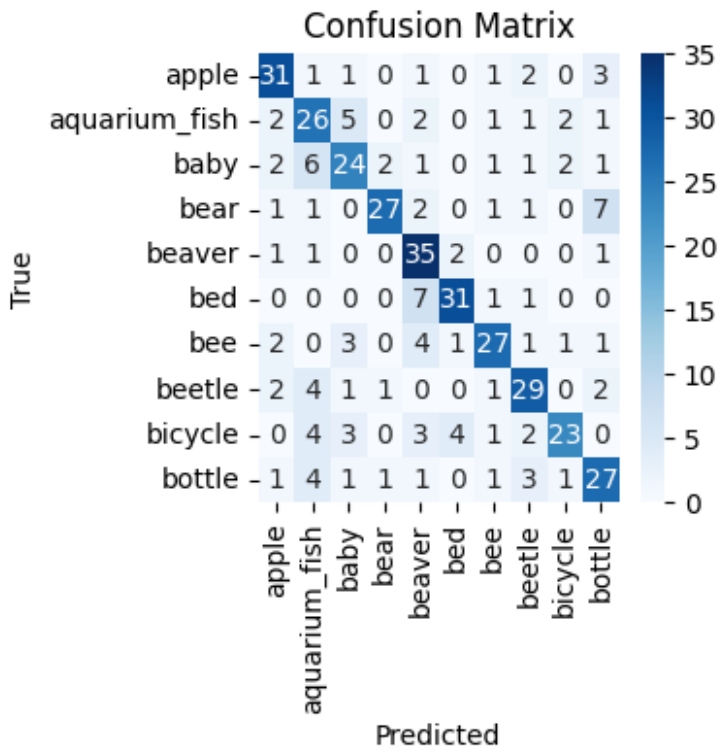
```
# Experiment 5: Batch size
model_5 = cnn(batch_size=64)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
```

```
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 26s 600ms/step - accuracy: 0.2536 - loss:
2.3263 - val_accuracy: 0.1075 - val_loss: 2.4512
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 635ms/step - accuracy: 0.4863 - loss:
1.5271 - val_accuracy: 0.1000 - val_loss: 2.6126
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 18s 554ms/step - accuracy: 0.5971 - loss:
1.1969 - val_accuracy: 0.1150 - val_loss: 2.8938
Epoch 4/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 19s 517ms/step - accuracy: 0.6607 - loss:
1.0099 - val_accuracy: 0.1050 - val_loss: 3.0973
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 535ms/step - accuracy: 0.7030 - loss:
0.8300 - val_accuracy: 0.1275 - val_loss: 3.2408
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 25s 666ms/step - accuracy: 0.7556 - loss:
0.7153 - val_accuracy: 0.1450 - val_loss: 3.2739
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 35s 498ms/step - accuracy: 0.7953 - loss:
0.6118 - val_accuracy: 0.1250 - val_loss: 3.2927
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 554ms/step - accuracy: 0.8227 - loss:
0.5539 - val_accuracy: 0.2250 - val_loss: 2.8939
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 19s 496ms/step - accuracy: 0.8281 - loss:
0.5082 - val_accuracy: 0.2375 - val_loss: 2.8132
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 23s 557ms/step - accuracy: 0.8949 - loss:
0.3726 - val_accuracy: 0.3225 - val_loss: 2.5337
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 18s 501ms/step - accuracy: 0.9121 - loss:
0.2992 - val_accuracy: 0.3750 - val_loss: 2.2124
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 512ms/step - accuracy: 0.9302 - loss:
0.2521 - val_accuracy: 0.3850 - val_loss: 2.1209
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 530ms/step - accuracy: 0.9516 - loss:
0.2037 - val_accuracy: 0.5075 - val_loss: 1.6191
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 521ms/step - accuracy: 0.9451 - loss:
0.2071 - val_accuracy: 0.5075 - val_loss: 1.6198
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 18s 550ms/step - accuracy: 0.9655 - loss:
0.1514 - val_accuracy: 0.6000 - val_loss: 1.2764
Epoch 16/20
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 512ms/step - accuracy: 0.9738 - loss:
0.1216 - val_accuracy: 0.6075 - val_loss: 1.3929
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 19s 486ms/step - accuracy: 0.9805 - loss:
0.1121 - val_accuracy: 0.6700 - val_loss: 1.1376
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 489ms/step - accuracy: 0.9838 - loss:
0.0887 - val_accuracy: 0.7000 - val_loss: 1.0577
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 517ms/step - accuracy: 0.9811 - loss:
0.0870 - val_accuracy: 0.6850 - val_loss: 1.1265
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 497ms/step - accuracy: 0.9799 - loss:
0.0887 - val_accuracy: 0.6975 - val_loss: 1.1118
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.7116 - loss:
1.0765
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step
Total training time: 422.66 seconds
Test accuracy: 0.7000
Test loss: 1.0577
```





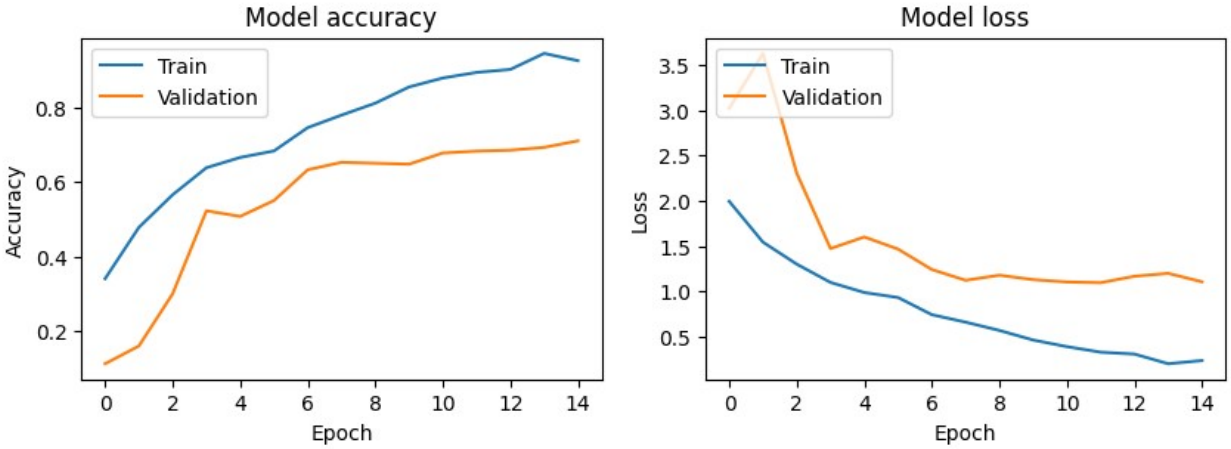
Total training time: 422.66 seconds Test accuracy: 0.7000 Test loss: 1.0577

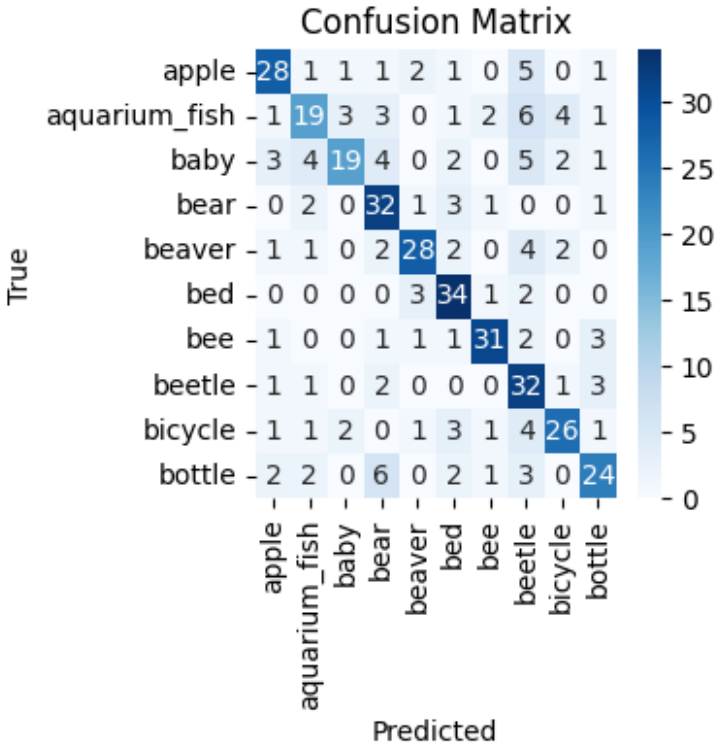
Better accuracy and loss but a little slower. Also clear signs of overfitting

```
cnn(batch_size=16)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 25s 149ms/step - accuracy: 0.2690 - loss:
2.2683 - val_accuracy: 0.1125 - val_loss: 3.0209
Epoch 2/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.4779 - loss:
1.5669 - val_accuracy: 0.1600 - val_loss: 3.6287
Epoch 3/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 139ms/step - accuracy: 0.5647 - loss:
1.2828 - val_accuracy: 0.3000 - val_loss: 2.3044
Epoch 4/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 147ms/step - accuracy: 0.6444 - loss:
1.0682 - val_accuracy: 0.5225 - val_loss: 1.4724
Epoch 5/20
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 140ms/step - accuracy: 0.6549 - loss:
0.9775 - val_accuracy: 0.5075 - val_loss: 1.6007
Epoch 6/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 136ms/step - accuracy: 0.6776 - loss:
0.9543 - val_accuracy: 0.5500 - val_loss: 1.4665
Epoch 7/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 18s 140ms/step - accuracy: 0.7375 - loss:
0.7474 - val_accuracy: 0.6325 - val_loss: 1.2409
Epoch 8/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 140ms/step - accuracy: 0.8029 - loss:
0.6217 - val_accuracy: 0.6525 - val_loss: 1.1219
Epoch 9/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 149ms/step - accuracy: 0.8246 - loss:
0.5509 - val_accuracy: 0.6500 - val_loss: 1.1781
Epoch 10/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 19s 137ms/step - accuracy: 0.8660 - loss:
0.4281 - val_accuracy: 0.6475 - val_loss: 1.1292
Epoch 11/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 138ms/step - accuracy: 0.8835 - loss:
0.3803 - val_accuracy: 0.6775 - val_loss: 1.1033
Epoch 12/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 150ms/step - accuracy: 0.9106 - loss:
0.2937 - val_accuracy: 0.6825 - val_loss: 1.0956
Epoch 13/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 19s 139ms/step - accuracy: 0.9137 - loss:
0.3013 - val_accuracy: 0.6850 - val_loss: 1.1666
Epoch 14/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 19s 149ms/step - accuracy: 0.9410 - loss:
0.2155 - val_accuracy: 0.6925 - val_loss: 1.1984
Epoch 15/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 141ms/step - accuracy: 0.9215 - loss:
0.2454 - val_accuracy: 0.7100 - val_loss: 1.1056
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 56ms/step - accuracy: 0.6497 - loss:
1.2218
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 80ms/step
Total training time: 304.09 seconds
Test accuracy: 0.6825
Test loss: 1.0956
```






```
<keras.src.callbacks.history.History at 0x7a1b8c3faf10>
```

Total training time: 304.09 seconds Test accuracy: 0.6825 Test loss: 1.0956

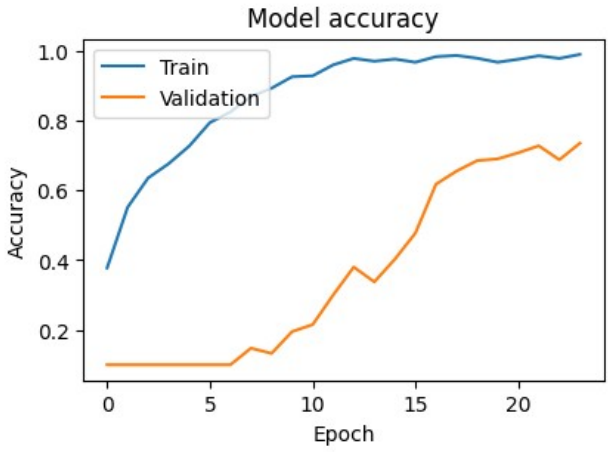
Little bit worse than baseline

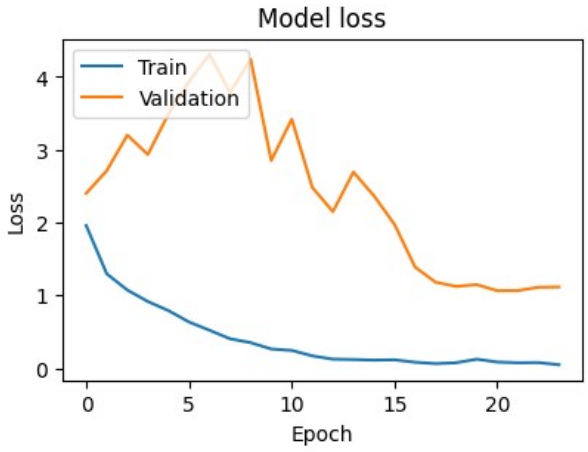
Experiment 6

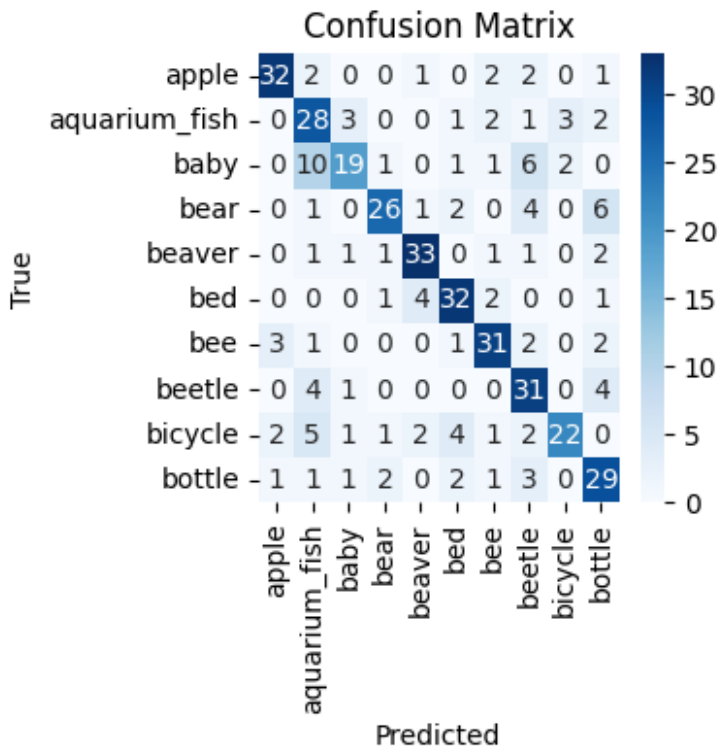
```
# Combining Swish and 64 batch size
cnn(activation='swish', batch_size=64, epochs=25)
```

```
Epoch 1/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 26s 630ms/step - accuracy: 0.3032 - loss:
2.2473 - val_accuracy: 0.1000 - val_loss: 2.4021
Epoch 2/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 538ms/step - accuracy: 0.5471 - loss:
1.2947 - val_accuracy: 0.1000 - val_loss: 2.7124
Epoch 3/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 552ms/step - accuracy: 0.6601 - loss:
1.0135 - val_accuracy: 0.1000 - val_loss: 3.2019
Epoch 4/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 521ms/step - accuracy: 0.6757 - loss:
0.9120 - val_accuracy: 0.1000 - val_loss: 2.9348
Epoch 5/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 583ms/step - accuracy: 0.7350 - loss:
0.7775 - val_accuracy: 0.1000 - val_loss: 3.4901
Epoch 6/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 19s 521ms/step - accuracy: 0.8048 - loss:
0.6212 - val_accuracy: 0.1000 - val_loss: 3.9370
Epoch 7/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 536ms/step - accuracy: 0.8342 - loss:
0.5073 - val_accuracy: 0.1000 - val_loss: 4.3095
Epoch 8/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 539ms/step - accuracy: 0.8714 - loss:
0.4051 - val_accuracy: 0.1475 - val_loss: 3.7844
Epoch 9/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 521ms/step - accuracy: 0.9011 - loss:
0.3325 - val_accuracy: 0.1325 - val_loss: 4.2434
Epoch 10/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 540ms/step - accuracy: 0.9321 - loss:
0.2537 - val_accuracy: 0.1950 - val_loss: 2.8517
Epoch 11/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 538ms/step - accuracy: 0.9368 - loss:
0.2284 - val_accuracy: 0.2150 - val_loss: 3.4197
Epoch 12/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 574ms/step - accuracy: 0.9592 - loss:
0.1738 - val_accuracy: 0.3000 - val_loss: 2.4819
Epoch 13/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 537ms/step - accuracy: 0.9799 - loss:
0.1173 - val_accuracy: 0.3800 - val_loss: 2.1506
Epoch 14/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 574ms/step - accuracy: 0.9751 - loss:
0.1014 - val_accuracy: 0.3375 - val_loss: 2.6930
Epoch 15/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 543ms/step - accuracy: 0.9818 - loss:
0.1025 - val_accuracy: 0.4025 - val_loss: 2.3677
Epoch 16/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 561ms/step - accuracy: 0.9695 - loss:
0.1121 - val_accuracy: 0.4775 - val_loss: 1.9742
Epoch 17/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 541ms/step - accuracy: 0.9807 - loss:
```

```
0.0893 - val_accuracy: 0.6175 - val_loss: 1.3887
Epoch 18/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 540ms/step - accuracy: 0.9877 - loss:
0.0592 - val_accuracy: 0.6550 - val_loss: 1.1802
Epoch 19/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 577ms/step - accuracy: 0.9839 - loss:
0.0610 - val_accuracy: 0.6850 - val_loss: 1.1237
Epoch 20/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 19s 539ms/step - accuracy: 0.9712 - loss:
0.1173 - val_accuracy: 0.6900 - val_loss: 1.1478
Epoch 21/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 521ms/step - accuracy: 0.9784 - loss:
0.0793 - val_accuracy: 0.7075 - val_loss: 1.0648
Epoch 22/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 555ms/step - accuracy: 0.9857 - loss:
0.0714 - val_accuracy: 0.7275 - val_loss: 1.0665
Epoch 23/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 537ms/step - accuracy: 0.9821 - loss:
0.0698 - val_accuracy: 0.6875 - val_loss: 1.1120
Epoch 24/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 543ms/step - accuracy: 0.9862 - loss:
0.0544 - val_accuracy: 0.7350 - val_loss: 1.1156
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 63ms/step - accuracy: 0.7112 - loss:
1.1025
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 81ms/step
Total training time: 480.65 seconds
Test accuracy: 0.7075
Test loss: 1.0648
```








```
<keras.src.callbacks.history.History at 0x7a1b83f3a590>
```

Total training time: 480.65 seconds Test accuracy: 0.7075 Test loss: 1.0648

Better accuracy, better loss, but not a huge improvement

Making small changes to the function

```
# Here I removed one convolutional layer from the stack

def cnn2(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):
```

```
    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        #model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        #model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Flatten and Dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        optimizer = optimizers.SGD(learning_rate=learning_rate)

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

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])
```

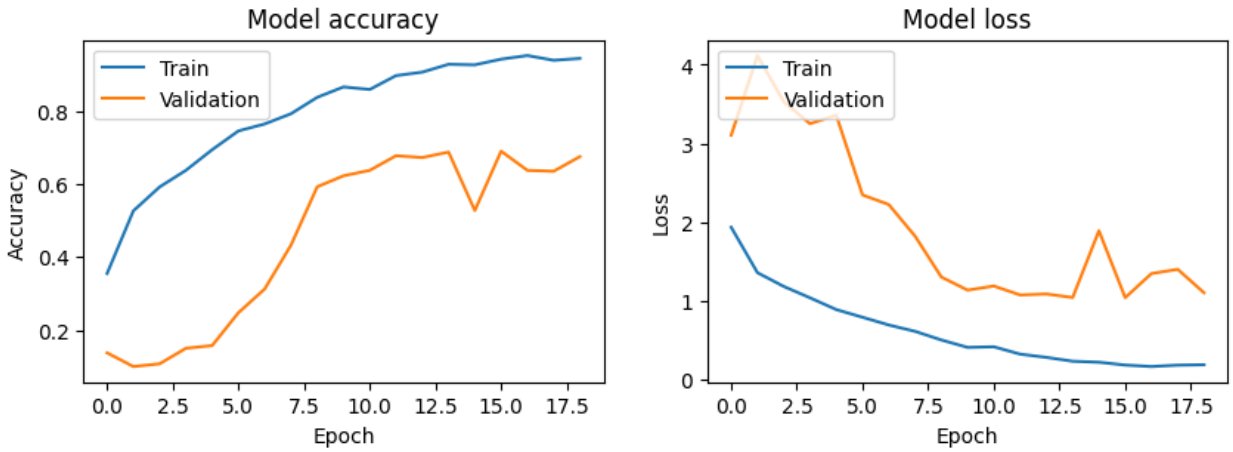
```
    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history
```

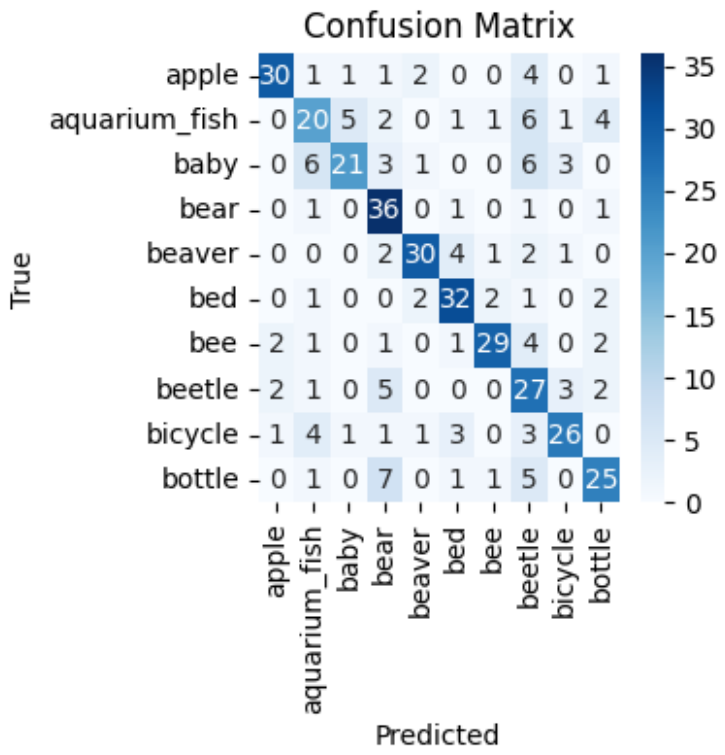
Testing new model configuration

```
model_b2 = cnn2()
model_c2 = cnn2(activation='elu')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 133ms/step - accuracy: 0.2946 - loss:
2.2170 - val_accuracy: 0.1375 - val_loss: 3.1070
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 126ms/step - accuracy: 0.5148 - loss:
1.3875 - val_accuracy: 0.1000 - val_loss: 4.1178
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.6069 - loss:
1.1637 - val_accuracy: 0.1075 - val_loss: 3.5317
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 134ms/step - accuracy: 0.6292 - loss:
1.0396 - val_accuracy: 0.1500 - val_loss: 3.2527
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 15s 213ms/step - accuracy: 0.6904 - loss:
0.9212 - val_accuracy: 0.1575 - val_loss: 3.3611
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 16s 140ms/step - accuracy: 0.7469 - loss:
0.7804 - val_accuracy: 0.2475 - val_loss: 2.3492
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 169ms/step - accuracy: 0.7840 - loss:
0.6659 - val_accuracy: 0.3125 - val_loss: 2.2269
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 176ms/step - accuracy: 0.7913 - loss:
0.6038 - val_accuracy: 0.4325 - val_loss: 1.8263
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 112ms/step - accuracy: 0.8595 - loss:
0.4676 - val_accuracy: 0.5925 - val_loss: 1.3075
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.8701 - loss:
0.3979 - val_accuracy: 0.6225 - val_loss: 1.1408
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 138ms/step - accuracy: 0.8671 - loss:
0.3922 - val_accuracy: 0.6375 - val_loss: 1.1934
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 128ms/step - accuracy: 0.9038 - loss:
0.3170 - val_accuracy: 0.6775 - val_loss: 1.0797
Epoch 13/20
```

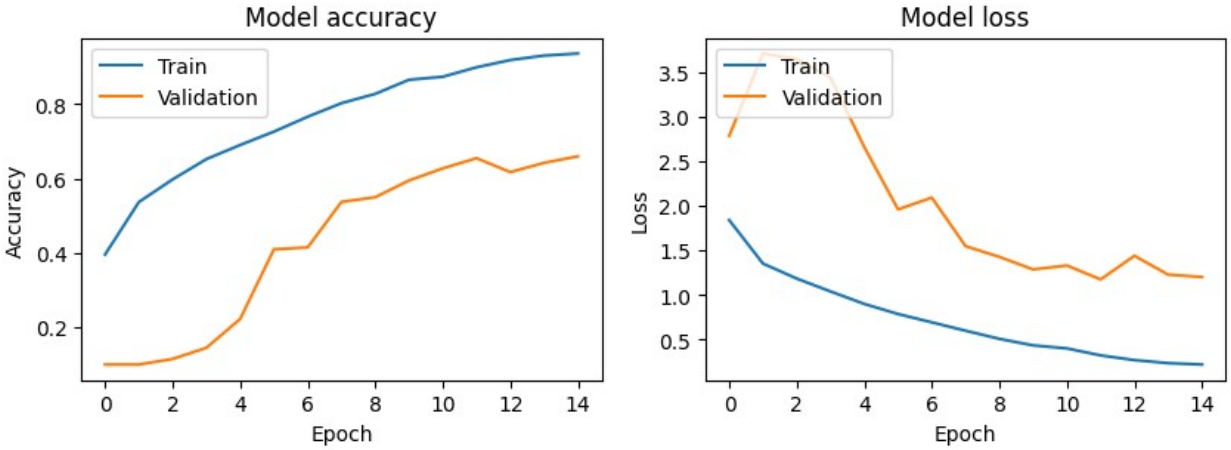
```
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 115ms/step - accuracy: 0.9210 - loss:
0.2668 - val_accuracy: 0.6725 - val_loss: 1.0924
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 136ms/step - accuracy: 0.9390 - loss:
0.2205 - val_accuracy: 0.6875 - val_loss: 1.0473
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 134ms/step - accuracy: 0.9464 - loss:
0.1982 - val_accuracy: 0.5275 - val_loss: 1.8950
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 114ms/step - accuracy: 0.9490 - loss:
0.1813 - val_accuracy: 0.6900 - val_loss: 1.0454
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.9474 - loss:
0.1707 - val_accuracy: 0.6375 - val_loss: 1.3513
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 118ms/step - accuracy: 0.9495 - loss:
0.1656 - val_accuracy: 0.6350 - val_loss: 1.4051
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 139ms/step - accuracy: 0.9561 - loss:
0.1699 - val_accuracy: 0.6750 - val_loss: 1.1078
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.6802 - loss:
1.1637
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step
Total training time: 202.72 seconds
Test accuracy: 0.6900
Test loss: 1.0454
```

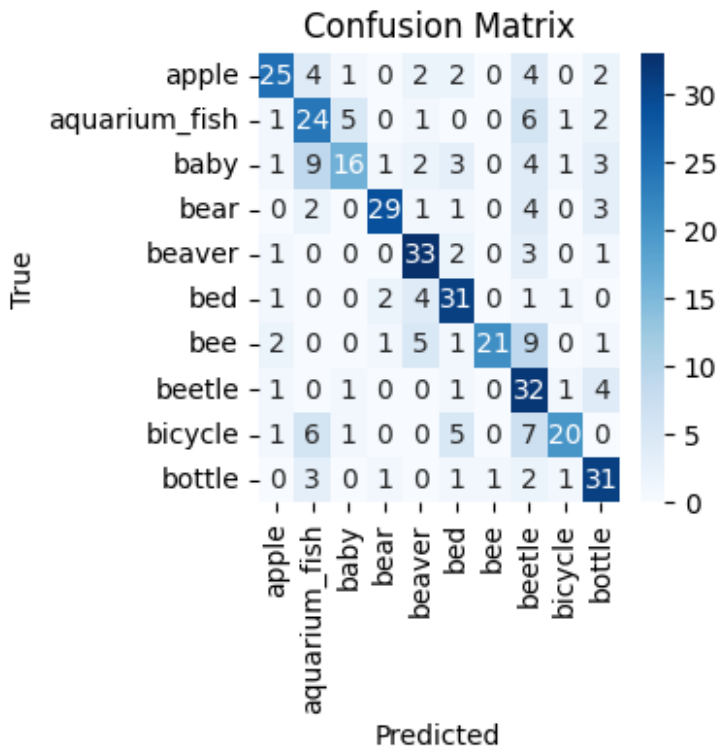





```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 14s 151ms/step - accuracy: 0.3047 - loss:
2.2037 - val_accuracy: 0.1000 - val_loss: 2.7895
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 144ms/step - accuracy: 0.5363 - loss:
1.3647 - val_accuracy: 0.1000 - val_loss: 3.7139
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 164ms/step - accuracy: 0.6137 - loss:
1.1484 - val_accuracy: 0.1150 - val_loss: 3.6522
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 137ms/step - accuracy: 0.6592 - loss:
1.0091 - val_accuracy: 0.1450 - val_loss: 3.4452
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 171ms/step - accuracy: 0.6878 - loss:
0.9241 - val_accuracy: 0.2225 - val_loss: 2.6640
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 133ms/step - accuracy: 0.7364 - loss:
0.7552 - val_accuracy: 0.4100 - val_loss: 1.9636
```

```
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 139ms/step - accuracy: 0.7889 - loss:
0.6611 - val_accuracy: 0.4150 - val_loss: 2.0959
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 136ms/step - accuracy: 0.8265 - loss:
0.5553 - val_accuracy: 0.5375 - val_loss: 1.5507
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 137ms/step - accuracy: 0.8332 - loss:
0.4996 - val_accuracy: 0.5500 - val_loss: 1.4302
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 125ms/step - accuracy: 0.8612 - loss:
0.4345 - val_accuracy: 0.5950 - val_loss: 1.2898
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 130ms/step - accuracy: 0.8733 - loss:
0.3975 - val_accuracy: 0.6275 - val_loss: 1.3333
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 148ms/step - accuracy: 0.8973 - loss:
0.3235 - val_accuracy: 0.6550 - val_loss: 1.1772
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 148ms/step - accuracy: 0.9175 - loss:
0.2588 - val_accuracy: 0.6175 - val_loss: 1.4427
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 128ms/step - accuracy: 0.9349 - loss:
0.2233 - val_accuracy: 0.6425 - val_loss: 1.2307
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 145ms/step - accuracy: 0.9348 - loss:
0.2158 - val_accuracy: 0.6600 - val_loss: 1.2048
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.6226 - loss:
1.3544
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step
Total training time: 170.99 seconds
Test accuracy: 0.6550
Test loss: 1.1772
```





Total training time: 202.72 seconds Test accuracy: 0.6900 Test loss: 1.0454

Better accuracy and loss, faster

Total training time: 170.99 seconds Test accuracy: 0.6550 Test loss: 1.1772

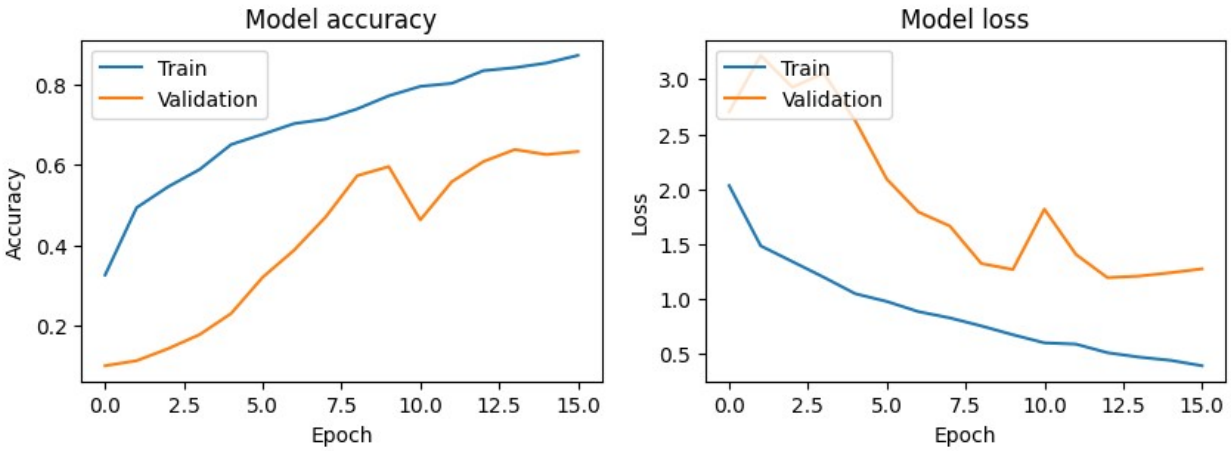
Elu wasn't any better.

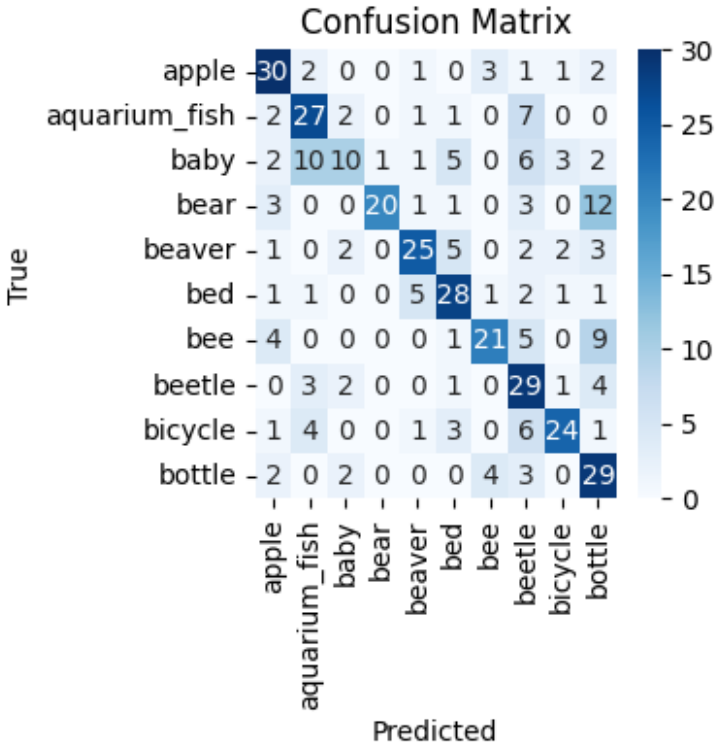
Model works pretty well with reduced layers also. I will stick with this one but try it with higher dropout to see if it doesn't overfit so badly. Also increasing epochs.

```
cnn2(dropout=0.3, epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 16s 139ms/step - accuracy: 0.2505 - loss:
2.3280 - val_accuracy: 0.1000 - val_loss: 2.7023
Epoch 2/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 125ms/step - accuracy: 0.4899 - loss:
1.5255 - val_accuracy: 0.1125 - val_loss: 3.2165
Epoch 3/25
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 135ms/step - accuracy: 0.5462 - loss:
1.3433 - val_accuracy: 0.1425 - val_loss: 2.9269
Epoch 4/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 121ms/step - accuracy: 0.5951 - loss:
1.1766 - val_accuracy: 0.1775 - val_loss: 3.0533
Epoch 5/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 135ms/step - accuracy: 0.6572 - loss:
1.0628 - val_accuracy: 0.2300 - val_loss: 2.6210
Epoch 6/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 123ms/step - accuracy: 0.6807 - loss:
0.9793 - val_accuracy: 0.3200 - val_loss: 2.0908
Epoch 7/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 111ms/step - accuracy: 0.7211 - loss:
0.8362 - val_accuracy: 0.3875 - val_loss: 1.7922
Epoch 8/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 143ms/step - accuracy: 0.7085 - loss:
0.8163 - val_accuracy: 0.4700 - val_loss: 1.6641
Epoch 9/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 138ms/step - accuracy: 0.7463 - loss:
0.7336 - val_accuracy: 0.5725 - val_loss: 1.3230
Epoch 10/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 130ms/step - accuracy: 0.7983 - loss:
0.6357 - val_accuracy: 0.5950 - val_loss: 1.2686
Epoch 11/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 137ms/step - accuracy: 0.8037 - loss:
0.5743 - val_accuracy: 0.4625 - val_loss: 1.8178
Epoch 12/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 139ms/step - accuracy: 0.7986 - loss:
0.5800 - val_accuracy: 0.5575 - val_loss: 1.4060
Epoch 13/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 131ms/step - accuracy: 0.8410 - loss:
0.4844 - val_accuracy: 0.6075 - val_loss: 1.1947
Epoch 14/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 107ms/step - accuracy: 0.8384 - loss:
0.4599 - val_accuracy: 0.6375 - val_loss: 1.2088
Epoch 15/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 106ms/step - accuracy: 0.8721 - loss:
0.4000 - val_accuracy: 0.6250 - val_loss: 1.2394
Epoch 16/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 130ms/step - accuracy: 0.8758 - loss:
0.3856 - val_accuracy: 0.6325 - val_loss: 1.2744
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6072 - loss:
1.2280
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step
Total training time: 159.58 seconds
Test accuracy: 0.6075
Test loss: 1.1947
```






```
<keras.src.callbacks.history.History at 0x7bd3cfc19f90>
```

Total training time: 159.58 seconds Test accuracy: 0.6075 Test loss: 1.1947

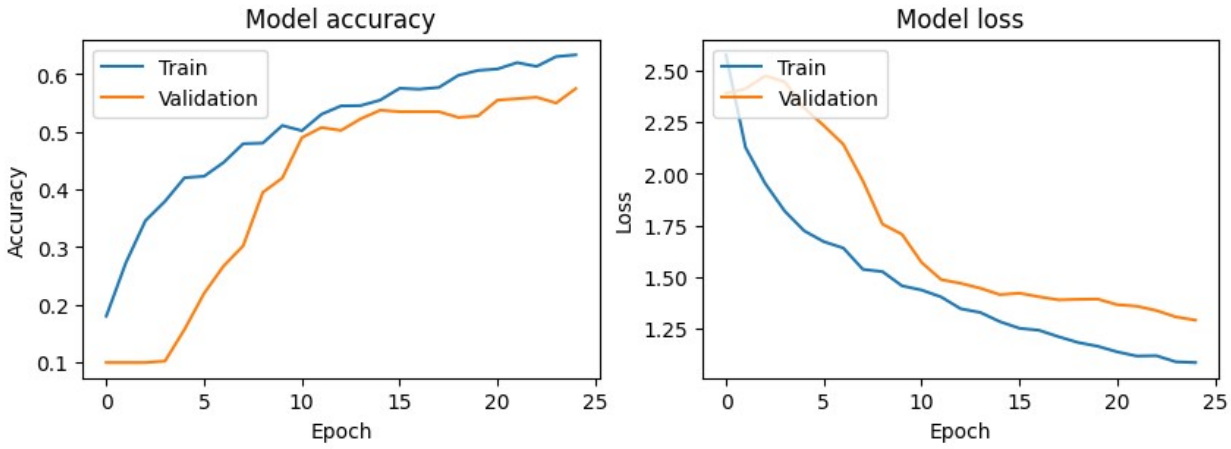
Didn't work as I hoped.

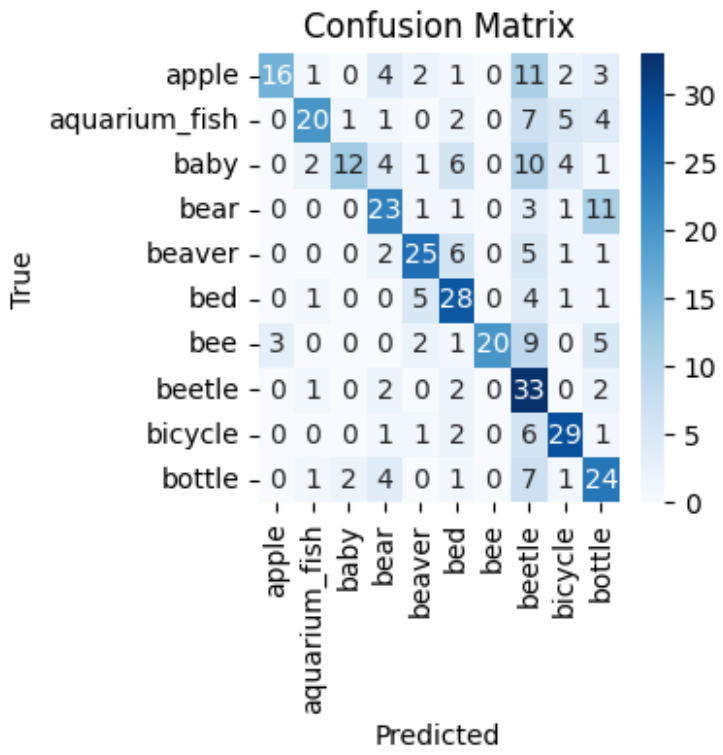
```
cnn2(dropout=0.3, epochs=25, learning_rate=0.0001)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
```

```
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 188ms/step - accuracy: 0.1410 - loss:
2.7670 - val_accuracy: 0.1000 - val_loss: 2.3907
Epoch 2/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 136ms/step - accuracy: 0.2515 - loss:
2.1628 - val_accuracy: 0.1000 - val_loss: 2.4120
Epoch 3/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 111ms/step - accuracy: 0.3436 - loss:
1.9361 - val_accuracy: 0.1000 - val_loss: 2.4751
Epoch 4/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 114ms/step - accuracy: 0.3723 - loss:
1.8070 - val_accuracy: 0.1025 - val_loss: 2.4466
Epoch 5/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 131ms/step - accuracy: 0.4222 - loss:
1.7489 - val_accuracy: 0.1575 - val_loss: 2.3186
Epoch 6/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 122ms/step - accuracy: 0.4311 - loss:
1.6701 - val_accuracy: 0.2200 - val_loss: 2.2338
Epoch 7/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 125ms/step - accuracy: 0.4314 - loss:
1.6845 - val_accuracy: 0.2675 - val_loss: 2.1428
Epoch 8/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 125ms/step - accuracy: 0.4798 - loss:
1.5217 - val_accuracy: 0.3025 - val_loss: 1.9653
Epoch 9/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 148ms/step - accuracy: 0.4741 - loss:
1.5380 - val_accuracy: 0.3950 - val_loss: 1.7561
Epoch 10/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 114ms/step - accuracy: 0.5270 - loss:
1.4662 - val_accuracy: 0.4200 - val_loss: 1.7063
Epoch 11/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 138ms/step - accuracy: 0.5006 - loss:
1.4534 - val_accuracy: 0.4900 - val_loss: 1.5700
Epoch 12/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 143ms/step - accuracy: 0.5221 - loss:
1.3988 - val_accuracy: 0.5075 - val_loss: 1.4867
Epoch 13/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 130ms/step - accuracy: 0.5361 - loss:
1.3681 - val_accuracy: 0.5025 - val_loss: 1.4688
Epoch 14/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 111ms/step - accuracy: 0.5496 - loss:
1.3113 - val_accuracy: 0.5225 - val_loss: 1.4451
Epoch 15/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 131ms/step - accuracy: 0.5550 - loss:
1.2606 - val_accuracy: 0.5375 - val_loss: 1.4140
Epoch 16/25
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 113ms/step - accuracy: 0.5883 - loss:
1.2120 - val_accuracy: 0.5350 - val_loss: 1.4210
Epoch 17/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 127ms/step - accuracy: 0.5811 - loss:
1.2373 - val_accuracy: 0.5350 - val_loss: 1.4037
Epoch 18/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 111ms/step - accuracy: 0.5867 - loss:
1.2465 - val_accuracy: 0.5350 - val_loss: 1.3885
Epoch 19/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 111ms/step - accuracy: 0.6003 - loss:
1.1655 - val_accuracy: 0.5250 - val_loss: 1.3910
Epoch 20/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 123ms/step - accuracy: 0.5897 - loss:
1.1999 - val_accuracy: 0.5275 - val_loss: 1.3924
Epoch 21/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 131ms/step - accuracy: 0.6164 - loss:
1.1168 - val_accuracy: 0.5550 - val_loss: 1.3653
Epoch 22/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 112ms/step - accuracy: 0.6176 - loss:
1.1217 - val_accuracy: 0.5575 - val_loss: 1.3580
Epoch 23/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 132ms/step - accuracy: 0.6140 - loss:
1.1180 - val_accuracy: 0.5600 - val_loss: 1.3370
Epoch 24/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 140ms/step - accuracy: 0.6347 - loss:
1.0530 - val_accuracy: 0.5500 - val_loss: 1.3058
Epoch 25/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 109ms/step - accuracy: 0.6328 - loss:
1.1032 - val_accuracy: 0.5750 - val_loss: 1.2904
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4977 - loss:
1.5787
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step
Total training time: 237.50 seconds
Test accuracy: 0.5750
Test loss: 1.2904
```






```
<keras.src.callbacks.history.History at 0x7bd3dec486d0>
```

Total training time: 237.50 seconds Test accuracy: 0.5750 Test loss: 1.2904

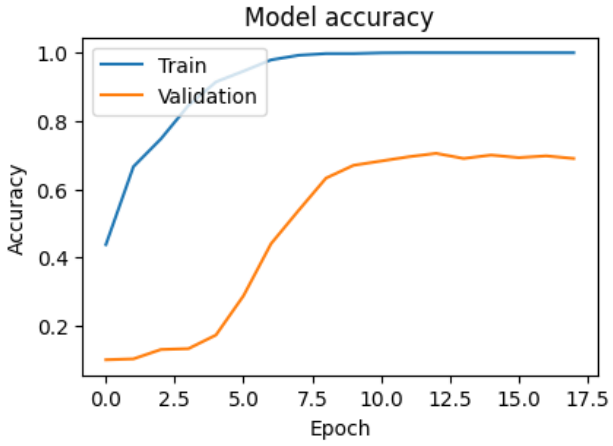
Smaller learning rate didn't help

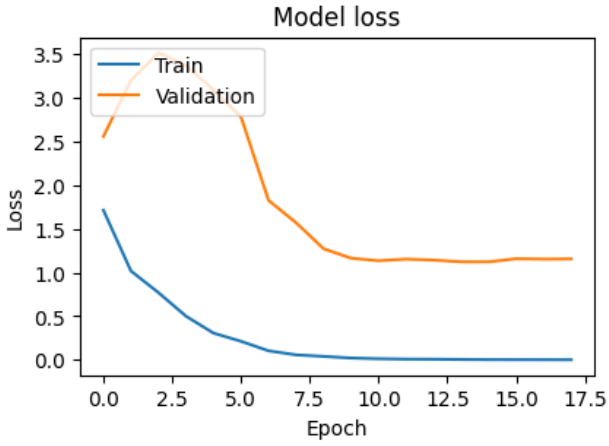
```
cnn2(dropout=0)
```

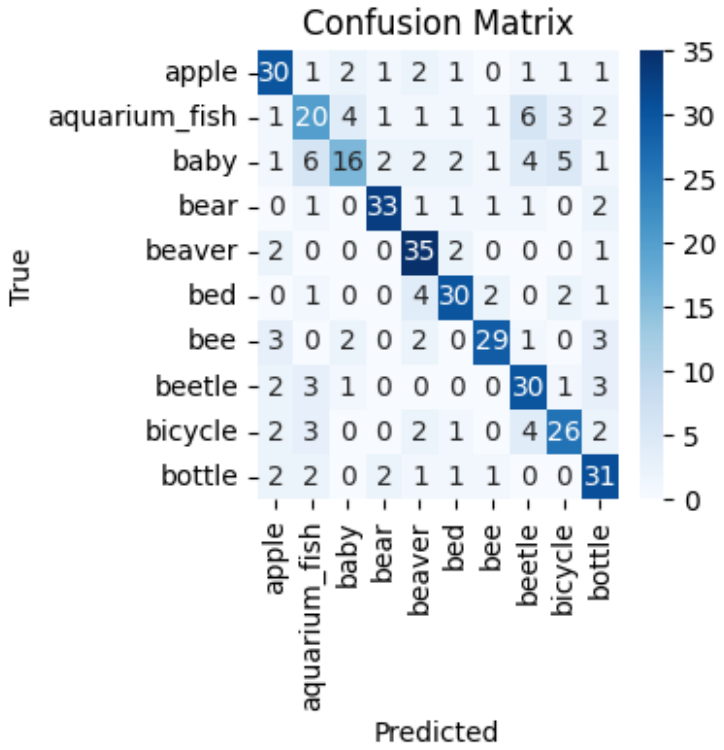
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 96ms/step - accuracy: 0.3308 - loss:
2.0812 - val_accuracy: 0.1000 - val_loss: 2.5592
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 99ms/step - accuracy: 0.6634 - loss:
1.0334 - val_accuracy: 0.1025 - val_loss: 3.2023
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 120ms/step - accuracy: 0.7370 - loss:
0.7666 - val_accuracy: 0.1300 - val_loss: 3.5168
Epoch 4/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 129ms/step - accuracy: 0.8589 - loss:
0.4769 - val_accuracy: 0.1325 - val_loss: 3.3698
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 122ms/step - accuracy: 0.9127 - loss:
0.3107 - val_accuracy: 0.1725 - val_loss: 3.0900
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 145ms/step - accuracy: 0.9479 - loss:
0.2078 - val_accuracy: 0.2875 - val_loss: 2.7819
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 135ms/step - accuracy: 0.9814 - loss:
0.1031 - val_accuracy: 0.4400 - val_loss: 1.8263
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 90ms/step - accuracy: 0.9933 - loss:
0.0576 - val_accuracy: 0.5375 - val_loss: 1.5712
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 91ms/step - accuracy: 0.9980 - loss:
0.0418 - val_accuracy: 0.6325 - val_loss: 1.2732
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.9973 - loss:
0.0224 - val_accuracy: 0.6700 - val_loss: 1.1650
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 123ms/step - accuracy: 0.9995 - loss:
0.0149 - val_accuracy: 0.6825 - val_loss: 1.1357
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 108ms/step - accuracy: 1.0000 - loss:
0.0088 - val_accuracy: 0.6950 - val_loss: 1.1541
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 89ms/step - accuracy: 1.0000 - loss:
0.0088 - val_accuracy: 0.7050 - val_loss: 1.1430
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 91ms/step - accuracy: 1.0000 - loss:
0.0068 - val_accuracy: 0.6900 - val_loss: 1.1235
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 122ms/step - accuracy: 1.0000 - loss:
0.0034 - val_accuracy: 0.7000 - val_loss: 1.1229
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 1.0000 - loss:
0.0031 - val_accuracy: 0.6925 - val_loss: 1.1587
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 90ms/step - accuracy: 1.0000 - loss:
0.0029 - val_accuracy: 0.6975 - val_loss: 1.1545
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 93ms/step - accuracy: 1.0000 - loss:
0.0024 - val_accuracy: 0.6900 - val_loss: 1.1571
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.6699 - loss:
1.2374
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
Total training time: 169.77 seconds
```

```
Test accuracy: 0.7000
Test loss: 1.1229
```








```
<keras.src.callbacks.history.History at 0x7befed34cf10>
```

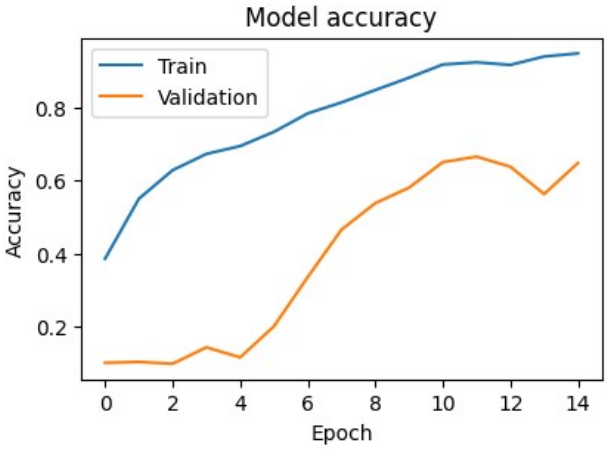
Total training time: 169.77 seconds Test accuracy: 0.7000 Test loss: 1.1229

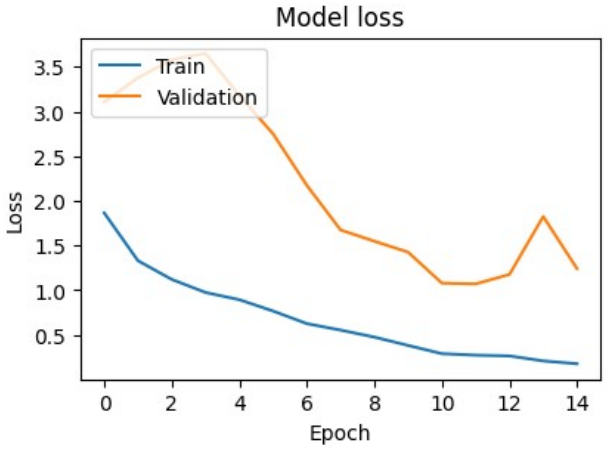
0 dropout has clear overfitting as can be expected and it doesn't perform that much better compared to 0.2

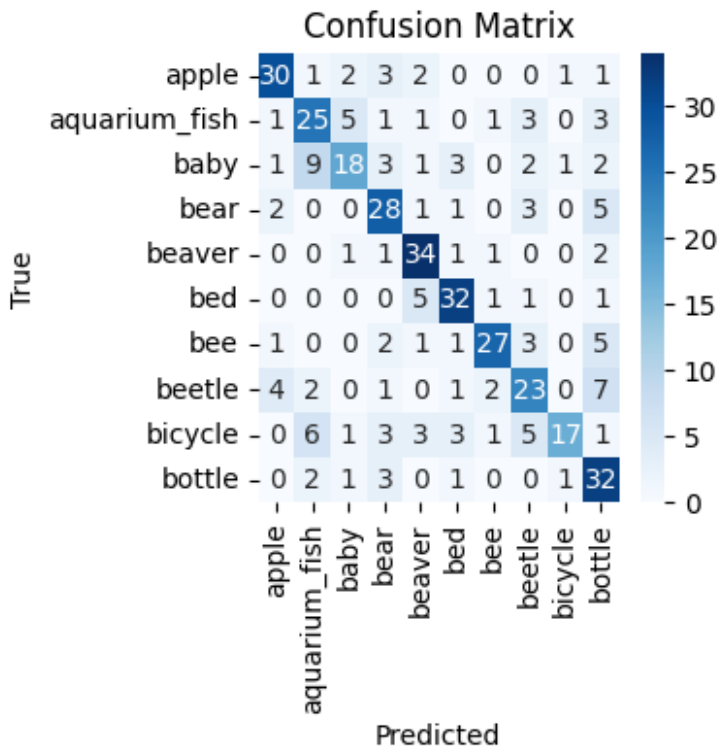
```
# Trying with swish to see if it still performs as well as before
cnn2(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 126ms/step - accuracy: 0.3159 - loss:
2.1802 - val_accuracy: 0.1000 - val_loss: 3.1079
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 101ms/step - accuracy: 0.5421 - loss:
1.3537 - val_accuracy: 0.1025 - val_loss: 3.3763
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 101ms/step - accuracy: 0.6112 - loss:
1.1757 - val_accuracy: 0.0975 - val_loss: 3.5795
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - accuracy: 0.6672 - loss:
0.9920 - val_accuracy: 0.1425 - val_loss: 3.6505
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 107ms/step - accuracy: 0.7038 - loss:
0.8625 - val_accuracy: 0.1150 - val_loss: 3.1803
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 101ms/step - accuracy: 0.7424 - loss:
0.7450 - val_accuracy: 0.2000 - val_loss: 2.7502
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 140ms/step - accuracy: 0.7972 - loss:
0.5922 - val_accuracy: 0.3350 - val_loss: 2.1738
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 123ms/step - accuracy: 0.8399 - loss:
0.5018 - val_accuracy: 0.4650 - val_loss: 1.6741
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 107ms/step - accuracy: 0.8566 - loss:
0.4504 - val_accuracy: 0.5375 - val_loss: 1.5480
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 106ms/step - accuracy: 0.8824 - loss:
0.3804 - val_accuracy: 0.5800 - val_loss: 1.4257
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 132ms/step - accuracy: 0.9284 - loss:
0.2838 - val_accuracy: 0.6500 - val_loss: 1.0786
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 118ms/step - accuracy: 0.9160 - loss:
0.2801 - val_accuracy: 0.6650 - val_loss: 1.0702
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 102ms/step - accuracy: 0.9279 - loss:
0.2358 - val_accuracy: 0.6375 - val_loss: 1.1763
```

```
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 124ms/step - accuracy: 0.9416 - loss:
0.2033 - val_accuracy: 0.5625 - val_loss: 1.8248
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 109ms/step - accuracy: 0.9452 - loss:
0.1801 - val_accuracy: 0.6475 - val_loss: 1.2418
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.6682 - loss:
1.1082
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step
Total training time: 131.18 seconds
Test accuracy: 0.6650
Test loss: 1.0702
```








```
<keras.src.callbacks.history.History at 0x7a1b86648490>
```

Total training time: 131.18 seconds Test accuracy: 0.6650 Test loss: 1.0702

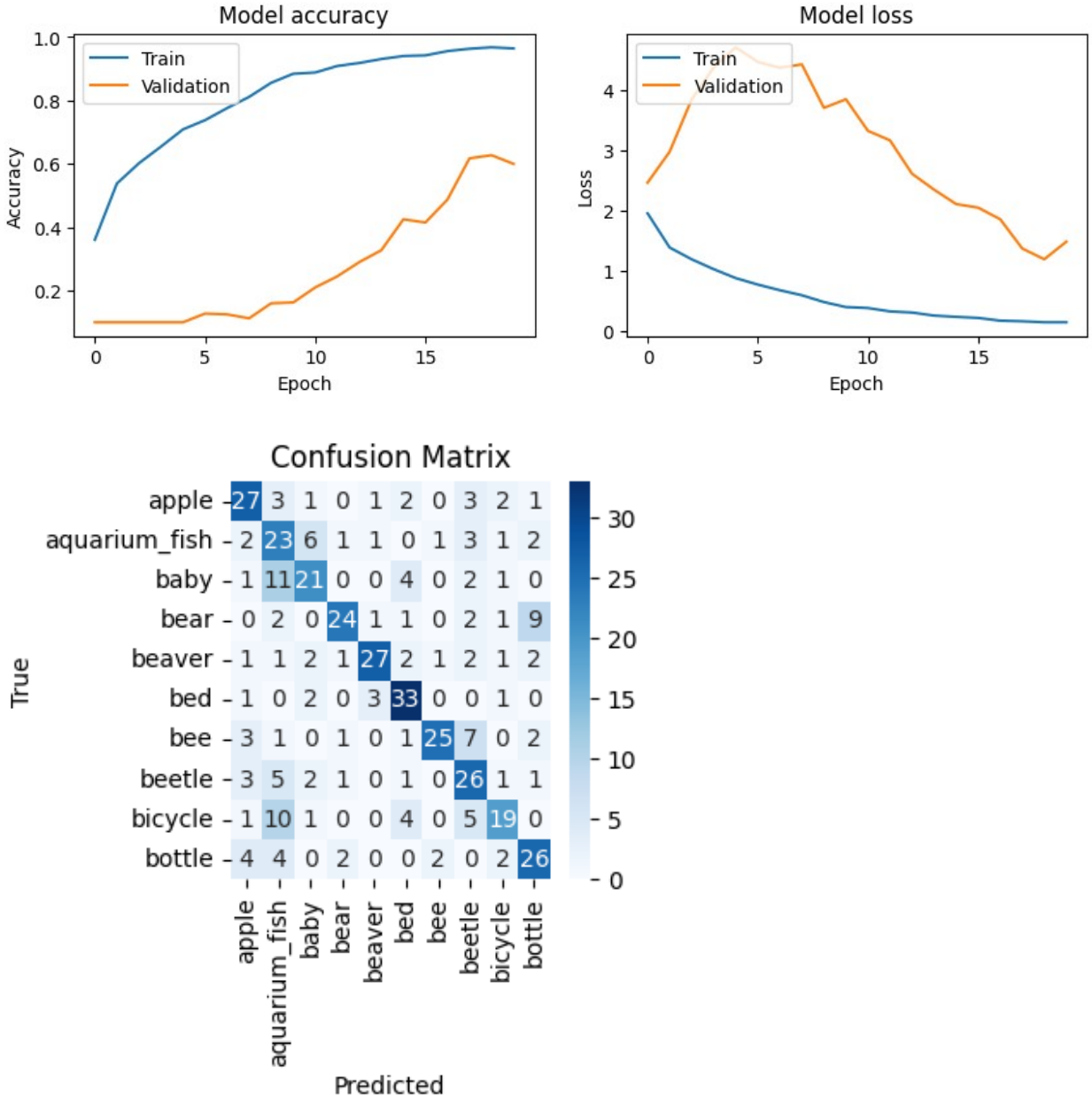
Accuracy drops but loss stays pretty much the same

```
cnn2(activation='swish', batch_size=64)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 215ms/step - accuracy: 0.2838 - loss:
2.2518 - val_accuracy: 0.1000 - val_loss: 2.4572
Epoch 2/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 203ms/step - accuracy: 0.5408 - loss:
1.4170 - val_accuracy: 0.1000 - val_loss: 2.9690
Epoch 3/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 250ms/step - accuracy: 0.5999 - loss:
1.2077 - val_accuracy: 0.1000 - val_loss: 3.8392
Epoch 4/20
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 201ms/step - accuracy: 0.6469 - loss:
1.0404 - val_accuracy: 0.1000 - val_loss: 4.3698
Epoch 5/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 211ms/step - accuracy: 0.7272 - loss:
0.8645 - val_accuracy: 0.1000 - val_loss: 4.7044
Epoch 6/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 200ms/step - accuracy: 0.7394 - loss:
0.7449 - val_accuracy: 0.1275 - val_loss: 4.4615
Epoch 7/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 236ms/step - accuracy: 0.7804 - loss:
0.6514 - val_accuracy: 0.1250 - val_loss: 4.3647
Epoch 8/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 200ms/step - accuracy: 0.8104 - loss:
0.5754 - val_accuracy: 0.1125 - val_loss: 4.4211
Epoch 9/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 239ms/step - accuracy: 0.8581 - loss:
0.4707 - val_accuracy: 0.1600 - val_loss: 3.6990
Epoch 10/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 269ms/step - accuracy: 0.8909 - loss:
0.3854 - val_accuracy: 0.1625 - val_loss: 3.8417
Epoch 11/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 201ms/step - accuracy: 0.9002 - loss:
0.3542 - val_accuracy: 0.2100 - val_loss: 3.3168
Epoch 12/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 201ms/step - accuracy: 0.9100 - loss:
0.3180 - val_accuracy: 0.2450 - val_loss: 3.1591
Epoch 13/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 233ms/step - accuracy: 0.9177 - loss:
0.3072 - val_accuracy: 0.2900 - val_loss: 2.6050
Epoch 14/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 227ms/step - accuracy: 0.9345 - loss:
0.2327 - val_accuracy: 0.3275 - val_loss: 2.3408
Epoch 15/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 200ms/step - accuracy: 0.9426 - loss:
0.2236 - val_accuracy: 0.4250 - val_loss: 2.1012
Epoch 16/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 199ms/step - accuracy: 0.9554 - loss:
0.1917 - val_accuracy: 0.4150 - val_loss: 2.0416
Epoch 17/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 199ms/step - accuracy: 0.9613 - loss:
0.1537 - val_accuracy: 0.4875 - val_loss: 1.8496
Epoch 18/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 239ms/step - accuracy: 0.9680 - loss:
0.1490 - val_accuracy: 0.6175 - val_loss: 1.3640
Epoch 19/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 212ms/step - accuracy: 0.9719 - loss:
0.1327 - val_accuracy: 0.6275 - val_loss: 1.1846
Epoch 20/20
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 198ms/step - accuracy: 0.9613 - loss:
```

```
0.1405 - val_accuracy: 0.6000 - val_loss: 1.4766
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6247 - loss:
1.2604
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step
Total training time: 188.31 seconds
Test accuracy: 0.6275
Test loss: 1.1846
```




```
<keras.src.callbacks.history.History at 0x7beffe062350>
```

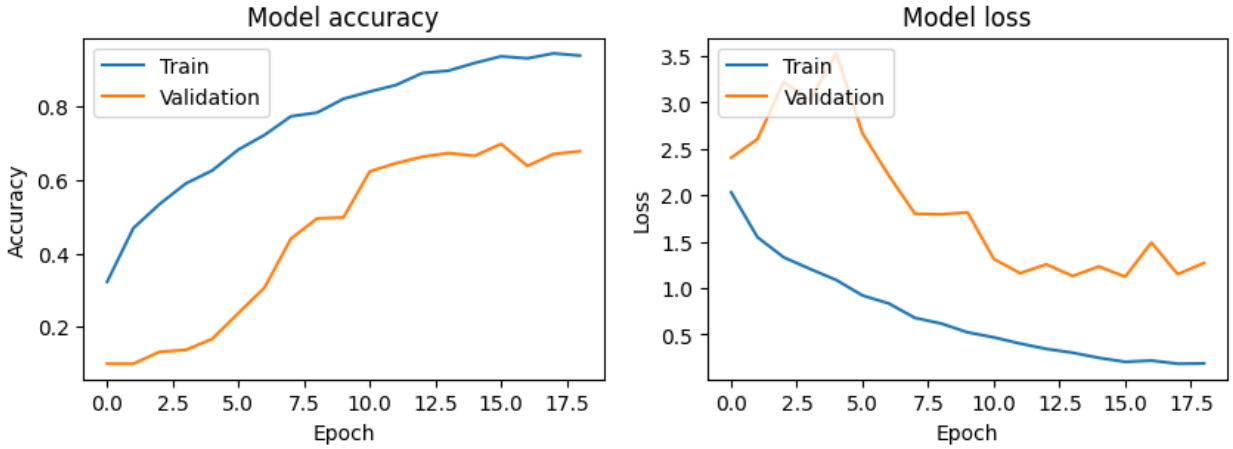
Total training time: 188.31 seconds Test accuracy: 0.6275 Test loss: 1.1846

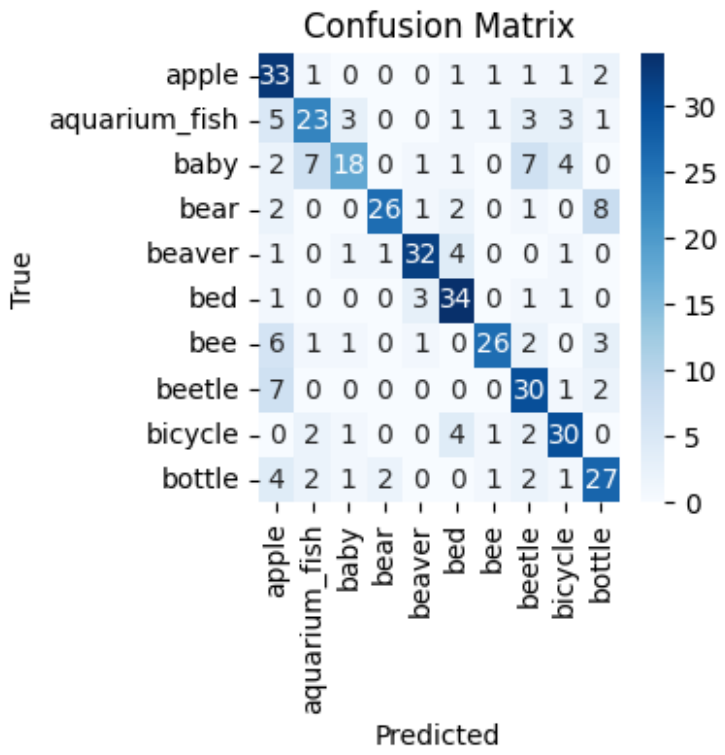
Increasing batch size with this model actually made it worse.

```
cnn2(activation='swish', conv_layers=5)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 35s 416ms/step - accuracy: 0.2764 - loss:
2.2194 - val_accuracy: 0.1000 - val_loss: 2.4051
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 387ms/step - accuracy: 0.4533 - loss:
1.5741 - val_accuracy: 0.1000 - val_loss: 2.6058
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 387ms/step - accuracy: 0.5377 - loss:
1.3180 - val_accuracy: 0.1325 - val_loss: 3.2236
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 381ms/step - accuracy: 0.5701 - loss:
1.2307 - val_accuracy: 0.1375 - val_loss: 3.0242
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 380ms/step - accuracy: 0.6147 - loss:
1.0952 - val_accuracy: 0.1675 - val_loss: 3.5297
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 376ms/step - accuracy: 0.6817 - loss:
0.9158 - val_accuracy: 0.2375 - val_loss: 2.6677
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 366ms/step - accuracy: 0.7454 - loss:
0.7809 - val_accuracy: 0.3075 - val_loss: 2.2143
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 43s 407ms/step - accuracy: 0.7828 - loss:
0.6301 - val_accuracy: 0.4400 - val_loss: 1.8002
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 390ms/step - accuracy: 0.7851 - loss:
0.6062 - val_accuracy: 0.4950 - val_loss: 1.7950
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 365ms/step - accuracy: 0.8267 - loss:
0.5207 - val_accuracy: 0.4975 - val_loss: 1.8161
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 391ms/step - accuracy: 0.8526 - loss:
0.4296 - val_accuracy: 0.6225 - val_loss: 1.3139
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 402ms/step - accuracy: 0.8669 - loss:
0.3889 - val_accuracy: 0.6450 - val_loss: 1.1609
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 395ms/step - accuracy: 0.8994 - loss:
```

```
0.3220 - val_accuracy: 0.6625 - val_loss: 1.2560
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 397ms/step - accuracy: 0.9042 - loss:
0.2810 - val_accuracy: 0.6725 - val_loss: 1.1304
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 391ms/step - accuracy: 0.9154 - loss:
0.2431 - val_accuracy: 0.6650 - val_loss: 1.2332
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 385ms/step - accuracy: 0.9414 - loss:
0.1968 - val_accuracy: 0.6975 - val_loss: 1.1226
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 407ms/step - accuracy: 0.9396 - loss:
0.1973 - val_accuracy: 0.6375 - val_loss: 1.4911
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 387ms/step - accuracy: 0.9374 - loss:
0.1788 - val_accuracy: 0.6700 - val_loss: 1.1502
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 382ms/step - accuracy: 0.9468 - loss:
0.1644 - val_accuracy: 0.6775 - val_loss: 1.2702
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.6852 - loss:
1.1868
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step
Total training time: 721.28 seconds
Test accuracy: 0.6975
Test loss: 1.1226
```






```
<keras.src.callbacks.history.History at 0x7befdb82a210>
```

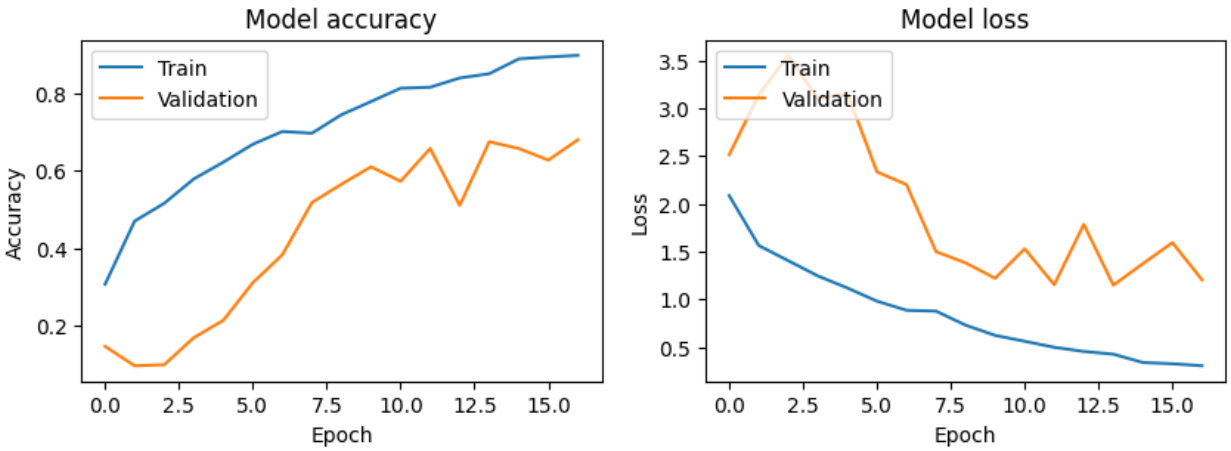
Total training time: 721.28 seconds Test accuracy: 0.6975 Test loss: 1.1226

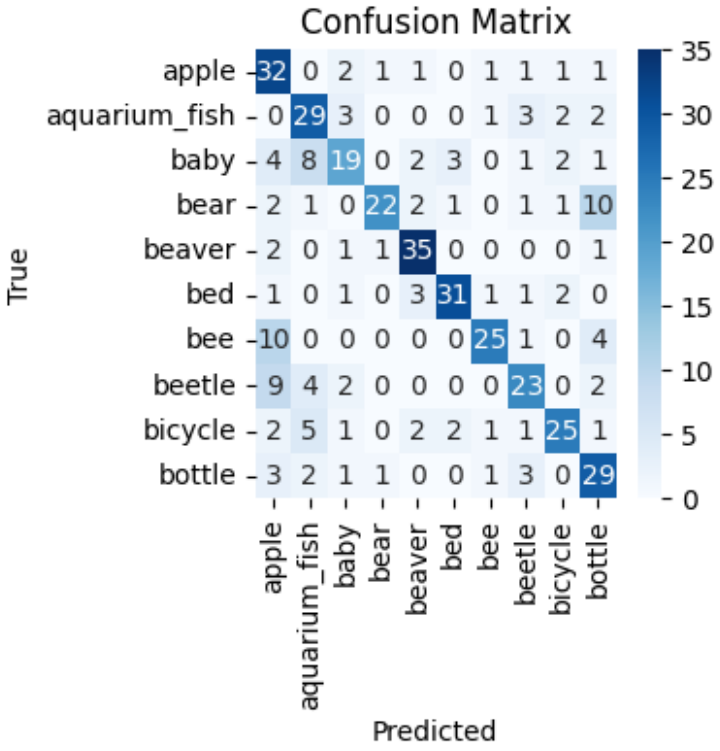
Swish doesn't work any better with more layers. The best version with swish is the one with three stacks of two convolutional layers

```
cnn2(conv_layers=5)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 36s 418ms/step - accuracy: 0.2361 - loss:
2.3743 - val_accuracy: 0.1450 - val_loss: 2.5125
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 376ms/step - accuracy: 0.4366 - loss:
1.6102 - val_accuracy: 0.0950 - val_loss: 3.1407
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 380ms/step - accuracy: 0.5115 - loss:
1.4010 - val_accuracy: 0.0975 - val_loss: 3.5521
```

```
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 372ms/step - accuracy: 0.5804 - loss:
1.2352 - val_accuracy: 0.1675 - val_loss: 3.0999
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 384ms/step - accuracy: 0.6220 - loss:
1.1147 - val_accuracy: 0.2125 - val_loss: 3.1557
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 390ms/step - accuracy: 0.6659 - loss:
0.9564 - val_accuracy: 0.3100 - val_loss: 2.3335
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 376ms/step - accuracy: 0.7167 - loss:
0.8655 - val_accuracy: 0.3825 - val_loss: 2.2009
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 383ms/step - accuracy: 0.6923 - loss:
0.8813 - val_accuracy: 0.5175 - val_loss: 1.4990
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 382ms/step - accuracy: 0.7408 - loss:
0.7236 - val_accuracy: 0.5650 - val_loss: 1.3820
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 369ms/step - accuracy: 0.7928 - loss:
0.5801 - val_accuracy: 0.6100 - val_loss: 1.2206
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 368ms/step - accuracy: 0.8238 - loss:
0.5450 - val_accuracy: 0.5725 - val_loss: 1.5290
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 379ms/step - accuracy: 0.8316 - loss:
0.4681 - val_accuracy: 0.6575 - val_loss: 1.1534
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 373ms/step - accuracy: 0.8388 - loss:
0.4649 - val_accuracy: 0.5100 - val_loss: 1.7847
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 373ms/step - accuracy: 0.8479 - loss:
0.4373 - val_accuracy: 0.6750 - val_loss: 1.1489
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 379ms/step - accuracy: 0.8927 - loss:
0.3319 - val_accuracy: 0.6575 - val_loss: 1.3735
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 375ms/step - accuracy: 0.9044 - loss:
0.2915 - val_accuracy: 0.6275 - val_loss: 1.5932
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 385ms/step - accuracy: 0.9091 - loss:
0.2689 - val_accuracy: 0.6800 - val_loss: 1.2046
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6909 - loss:
1.0715
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step
Total training time: 640.03 seconds
Test accuracy: 0.6750
Test loss: 1.1489
```






```
<keras.src.callbacks.history.History at 0x7befd9a5b650>
```

Total training time: 640.03 seconds Test accuracy: 0.6750 Test loss: 1.1489

Swish was better than relu with more layers.

Third iteration of the model

```
# Here I added learning rate scheduler when using sgd

def cnn3(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):

    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        #model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        #model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Flatten and Dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
```

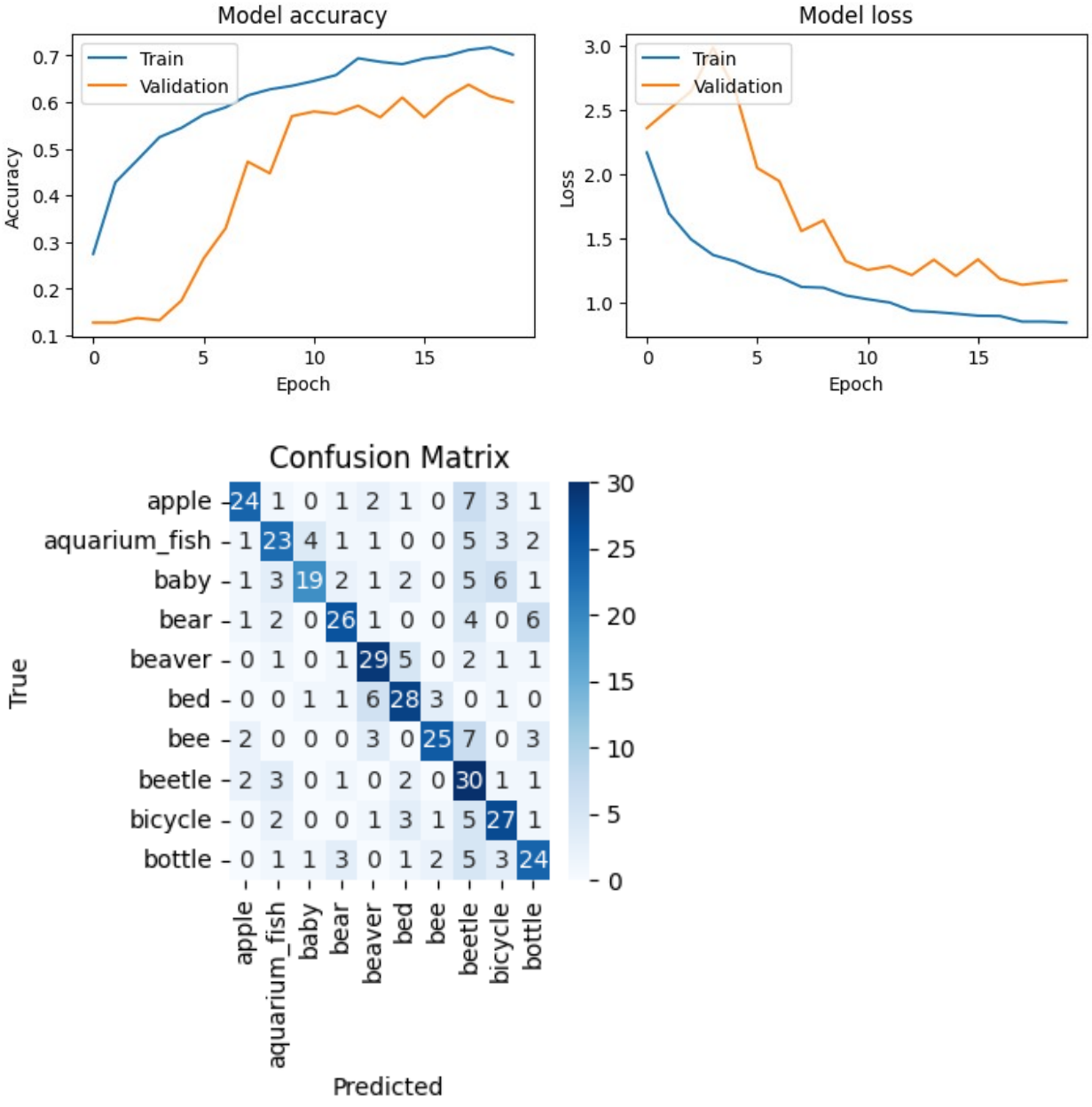
```
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)


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

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')
```

```
    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn3(optimizer_name='sgd', learning_rate=0.01)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 119ms/step - accuracy: 0.1977 - loss:
2.4751 - val_accuracy: 0.1275 - val_loss: 2.3614
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 130ms/step - accuracy: 0.4077 - loss:
1.7518 - val_accuracy: 0.1275 - val_loss: 2.5069
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 146ms/step - accuracy: 0.4767 - loss:
1.4958 - val_accuracy: 0.1375 - val_loss: 2.6470
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 112ms/step - accuracy: 0.5297 - loss:
1.3510 - val_accuracy: 0.1325 - val_loss: 2.9894
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 128ms/step - accuracy: 0.5359 - loss:
1.3513 - val_accuracy: 0.1750 - val_loss: 2.6522
Epoch 6/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 133ms/step - accuracy: 0.5642 - loss:
1.2821 - val_accuracy: 0.2650 - val_loss: 2.0507
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 138ms/step - accuracy: 0.6202 - loss:
1.1740 - val_accuracy: 0.3300 - val_loss: 1.9486
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 106ms/step - accuracy: 0.6283 - loss:
1.0988 - val_accuracy: 0.4725 - val_loss: 1.5602
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 106ms/step - accuracy: 0.6204 - loss:
1.1109 - val_accuracy: 0.4475 - val_loss: 1.6437
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 110ms/step - accuracy: 0.6374 - loss:
1.0608 - val_accuracy: 0.5700 - val_loss: 1.3268
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 130ms/step - accuracy: 0.6596 - loss:
0.9923 - val_accuracy: 0.5800 - val_loss: 1.2574
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 131ms/step - accuracy: 0.6630 - loss:
0.9887 - val_accuracy: 0.5750 - val_loss: 1.2873
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 130ms/step - accuracy: 0.6899 - loss:
0.9394 - val_accuracy: 0.5925 - val_loss: 1.2175
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 106ms/step - accuracy: 0.6890 - loss:
0.9343 - val_accuracy: 0.5675 - val_loss: 1.3369
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 12s 134ms/step - accuracy: 0.6753 - loss:
0.9290 - val_accuracy: 0.6100 - val_loss: 1.2104
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 131ms/step - accuracy: 0.7106 - loss:
0.8595 - val_accuracy: 0.5675 - val_loss: 1.3393
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 106ms/step - accuracy: 0.6898 - loss:
0.9185 - val_accuracy: 0.6100 - val_loss: 1.1887
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 129ms/step - accuracy: 0.7236 - loss:
0.8523 - val_accuracy: 0.6375 - val_loss: 1.1418
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.7320 - loss:
0.8118 - val_accuracy: 0.6125 - val_loss: 1.1613
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 108ms/step - accuracy: 0.6920 - loss:
0.8507 - val_accuracy: 0.6000 - val_loss: 1.1753
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6144 - loss:
1.2366
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step
Total training time: 194.39 seconds
```

```
Test accuracy: 0.6375
Test loss: 1.1418
```



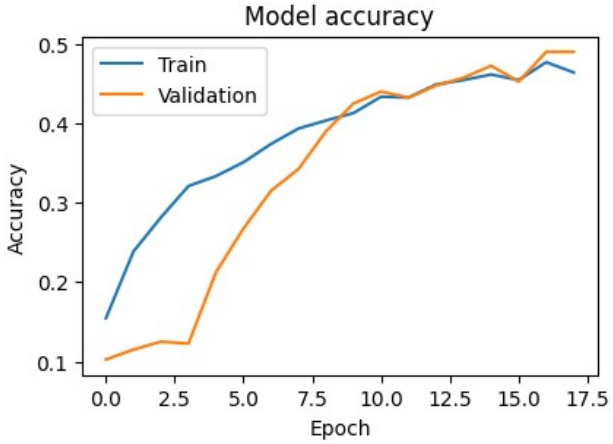

```
<keras.src.callbacks.history.History at 0x7bd3b68d8c90>
```

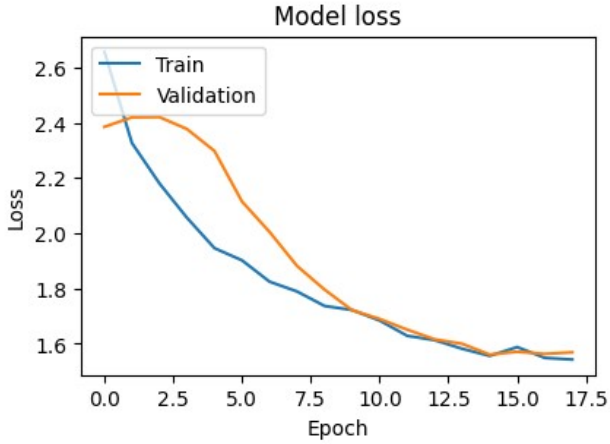
Total training time: 194.39 seconds Test accuracy: 0.6375 Test loss: 1.1418

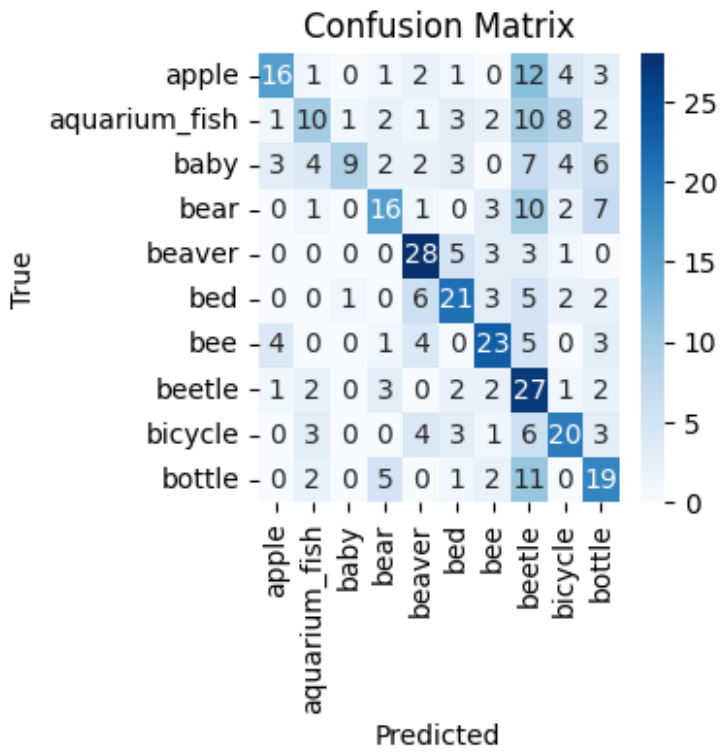
```
cnn3(optimizer_name='sgd', decay_rate=0.95)
cnn3(optimizer_name='sgd', decay_rate=0.85)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 133ms/step - accuracy: 0.1365 - loss:
2.7560 - val_accuracy: 0.1025 - val_loss: 2.3854
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 93ms/step - accuracy: 0.2088 - loss:
2.4131 - val_accuracy: 0.1150 - val_loss: 2.4199
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.2742 - loss:
2.1913 - val_accuracy: 0.1250 - val_loss: 2.4208
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 91ms/step - accuracy: 0.3022 - loss:
2.0579 - val_accuracy: 0.1225 - val_loss: 2.3775
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 91ms/step - accuracy: 0.3451 - loss:
1.9181 - val_accuracy: 0.2125 - val_loss: 2.2979
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.3346 - loss:
1.9067 - val_accuracy: 0.2675 - val_loss: 2.1150
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 91ms/step - accuracy: 0.3890 - loss:
1.7957 - val_accuracy: 0.3150 - val_loss: 2.0050
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.3948 - loss:
1.7805 - val_accuracy: 0.3425 - val_loss: 1.8816
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 91ms/step - accuracy: 0.4092 - loss:
1.7207 - val_accuracy: 0.3900 - val_loss: 1.7959
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 97ms/step - accuracy: 0.4072 - loss:
1.7207 - val_accuracy: 0.4250 - val_loss: 1.7215
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 104ms/step - accuracy: 0.4210 - loss:
1.7141 - val_accuracy: 0.4400 - val_loss: 1.6898
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 91ms/step - accuracy: 0.4393 - loss:
1.6372 - val_accuracy: 0.4325 - val_loss: 1.6511
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 120ms/step - accuracy: 0.4728 - loss:
1.5873 - val_accuracy: 0.4475 - val_loss: 1.6159
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 108ms/step - accuracy: 0.4661 - loss:
1.5713 - val_accuracy: 0.4575 - val_loss: 1.5998
```

```
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 91ms/step - accuracy: 0.4602 - loss:
1.5607 - val_accuracy: 0.4725 - val_loss: 1.5607
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 98ms/step - accuracy: 0.4417 - loss:
1.6376 - val_accuracy: 0.4525 - val_loss: 1.5709
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 107ms/step - accuracy: 0.4863 - loss:
1.5123 - val_accuracy: 0.4900 - val_loss: 1.5631
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 108ms/step - accuracy: 0.4642 - loss:
1.5421 - val_accuracy: 0.4900 - val_loss: 1.5689
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.4021 - loss:
1.7471
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step
Total training time: 148.51 seconds
Test accuracy: 0.4725
Test loss: 1.5607
```

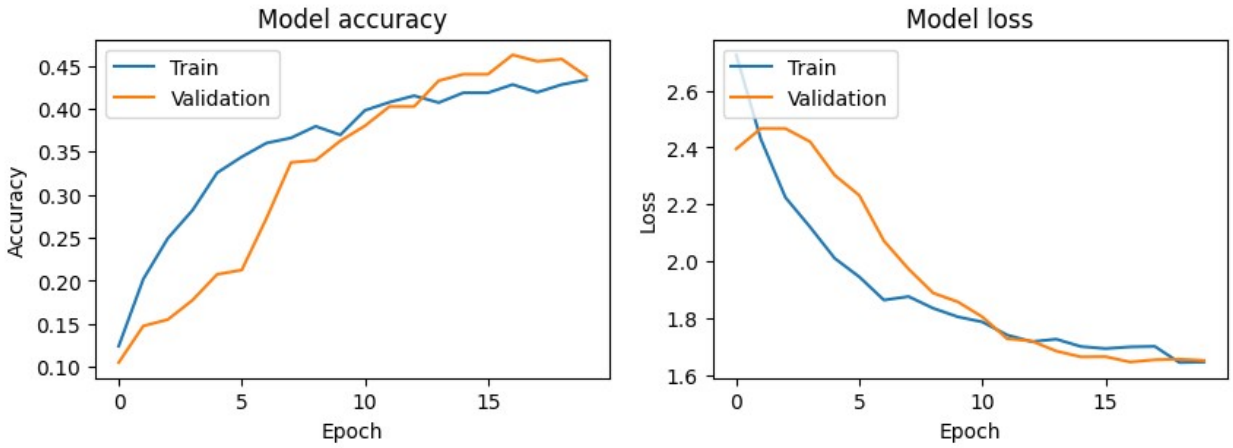


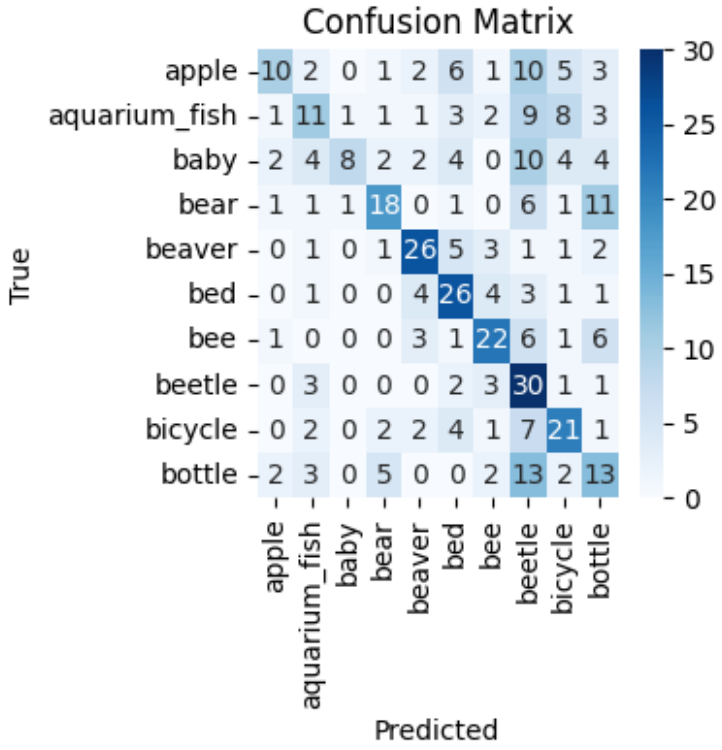





```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 8s 101ms/step - accuracy: 0.0930 - loss:
2.8423 - val_accuracy: 0.1050 - val_loss: 2.3947
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 105ms/step - accuracy: 0.1915 - loss:
2.4862 - val_accuracy: 0.1475 - val_loss: 2.4667
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 98ms/step - accuracy: 0.2283 - loss:
2.2874 - val_accuracy: 0.1550 - val_loss: 2.4664
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 91ms/step - accuracy: 0.2914 - loss:
2.1141 - val_accuracy: 0.1775 - val_loss: 2.4203
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 92ms/step - accuracy: 0.3232 - loss:
2.0182 - val_accuracy: 0.2075 - val_loss: 2.3026
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 92ms/step - accuracy: 0.3355 - loss:
1.9461 - val_accuracy: 0.2125 - val_loss: 2.2316
```

```
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.3727 - loss:
1.8546 - val_accuracy: 0.2725 - val_loss: 2.0718
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 92ms/step - accuracy: 0.3789 - loss:
1.8424 - val_accuracy: 0.3375 - val_loss: 1.9731
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 91ms/step - accuracy: 0.3699 - loss:
1.8514 - val_accuracy: 0.3400 - val_loss: 1.8887
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.3663 - loss:
1.8107 - val_accuracy: 0.3625 - val_loss: 1.8578
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 110ms/step - accuracy: 0.4111 - loss:
1.7679 - val_accuracy: 0.3800 - val_loss: 1.8047
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 9s 91ms/step - accuracy: 0.4163 - loss:
1.7199 - val_accuracy: 0.4025 - val_loss: 1.7284
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.3956 - loss:
1.7522 - val_accuracy: 0.4025 - val_loss: 1.7199
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 109ms/step - accuracy: 0.4043 - loss:
1.7298 - val_accuracy: 0.4325 - val_loss: 1.6848
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 91ms/step - accuracy: 0.4300 - loss:
1.6512 - val_accuracy: 0.4400 - val_loss: 1.6644
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 110ms/step - accuracy: 0.4139 - loss:
1.7152 - val_accuracy: 0.4400 - val_loss: 1.6652
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 101ms/step - accuracy: 0.4499 - loss:
1.6826 - val_accuracy: 0.4625 - val_loss: 1.6464
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 10s 92ms/step - accuracy: 0.4182 - loss:
1.6909 - val_accuracy: 0.4550 - val_loss: 1.6543
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 104ms/step - accuracy: 0.4101 - loss:
1.6905 - val_accuracy: 0.4575 - val_loss: 1.6562
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.4361 - loss:
1.6397 - val_accuracy: 0.4375 - val_loss: 1.6519
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.3748 - loss:
1.8996
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step
Total training time: 174.12 seconds
Test accuracy: 0.4625
Test loss: 1.6464
```






```
<keras.src.callbacks.history.History at 0x7a1ba041a490>
```

Total training time: 148.51 seconds Test accuracy: 0.4725 Test loss: 1.5607

Total training time: 174.12 seconds Test accuracy: 0.4625 Test loss: 1.6464

SGD just doesn't seem to be the number one choice for this model.

Fourth iteration of the model

```
# I'll try with added convolutional layers also
```

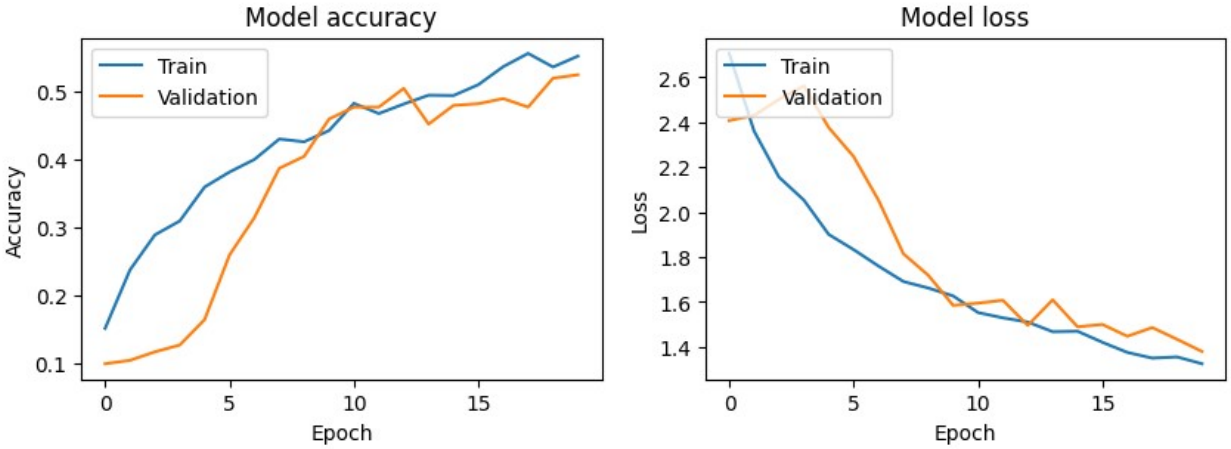
```
def cnn4(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):

    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Flatten and Dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)
```

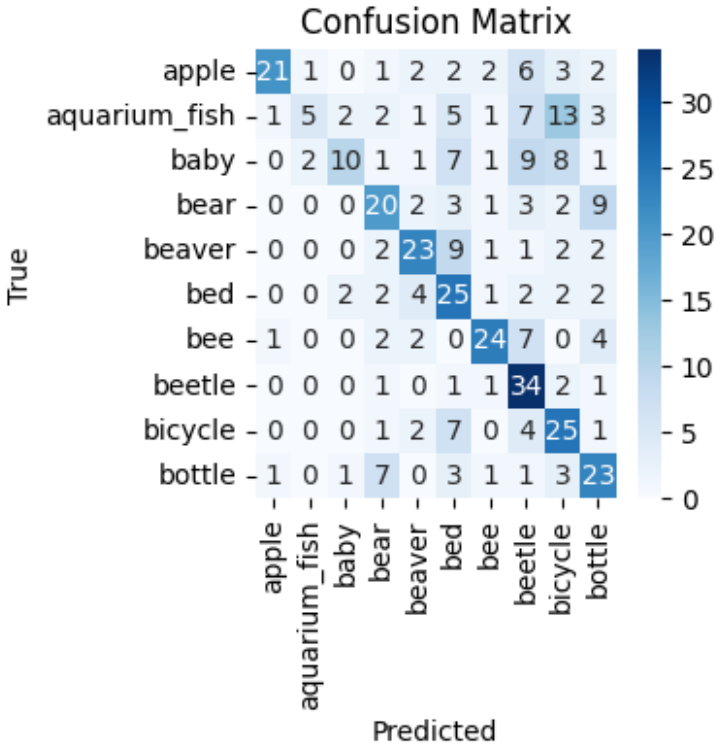
```
    # Compile
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
```

```
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn4(optimizer_name='sgd', decay_rate=0.95)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 277ms/step - accuracy: 0.1219 - loss:
2.8442 - val_accuracy: 0.1000 - val_loss: 2.4073
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 16s 262ms/step - accuracy: 0.2341 - loss:
2.3690 - val_accuracy: 0.1050 - val_loss: 2.4304
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 265ms/step - accuracy: 0.3029 - loss:
2.1362 - val_accuracy: 0.1175 - val_loss: 2.5016
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 435ms/step - accuracy: 0.2898 - loss:
2.1145 - val_accuracy: 0.1275 - val_loss: 2.5622
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 30s 267ms/step - accuracy: 0.3449 - loss:
1.9233 - val_accuracy: 0.1650 - val_loss: 2.3763
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 285ms/step - accuracy: 0.3777 - loss:
1.8503 - val_accuracy: 0.2600 - val_loss: 2.2474
Epoch 7/20
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 258ms/step - accuracy: 0.4154 - loss:
1.7184 - val_accuracy: 0.3150 - val_loss: 2.0531
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 268ms/step - accuracy: 0.4246 - loss:
1.6694 - val_accuracy: 0.3875 - val_loss: 1.8148
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 280ms/step - accuracy: 0.4423 - loss:
1.6280 - val_accuracy: 0.4050 - val_loss: 1.7191
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 265ms/step - accuracy: 0.4347 - loss:
1.6650 - val_accuracy: 0.4600 - val_loss: 1.5850
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 266ms/step - accuracy: 0.4657 - loss:
1.5869 - val_accuracy: 0.4775 - val_loss: 1.5950
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.4909 - loss:
1.4874 - val_accuracy: 0.4775 - val_loss: 1.6077
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 266ms/step - accuracy: 0.4973 - loss:
1.5223 - val_accuracy: 0.5050 - val_loss: 1.4961
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 288ms/step - accuracy: 0.4985 - loss:
1.4775 - val_accuracy: 0.4525 - val_loss: 1.6098
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 265ms/step - accuracy: 0.4936 - loss:
1.4578 - val_accuracy: 0.4800 - val_loss: 1.4896
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 304ms/step - accuracy: 0.5126 - loss:
1.4210 - val_accuracy: 0.4825 - val_loss: 1.5003
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 266ms/step - accuracy: 0.5223 - loss:
1.4047 - val_accuracy: 0.4900 - val_loss: 1.4476
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 285ms/step - accuracy: 0.5361 - loss:
1.4039 - val_accuracy: 0.4775 - val_loss: 1.4864
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 266ms/step - accuracy: 0.5369 - loss:
1.3625 - val_accuracy: 0.5200 - val_loss: 1.4341
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 283ms/step - accuracy: 0.5572 - loss:
1.3287 - val_accuracy: 0.5250 - val_loss: 1.3801
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.4410 - loss:
1.6077
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 76ms/step
Total training time: 403.90 seconds
Test accuracy: 0.5250
Test loss: 1.3801
```






```
<keras.src.callbacks.history.History at 0x7bf0a907a610>
```

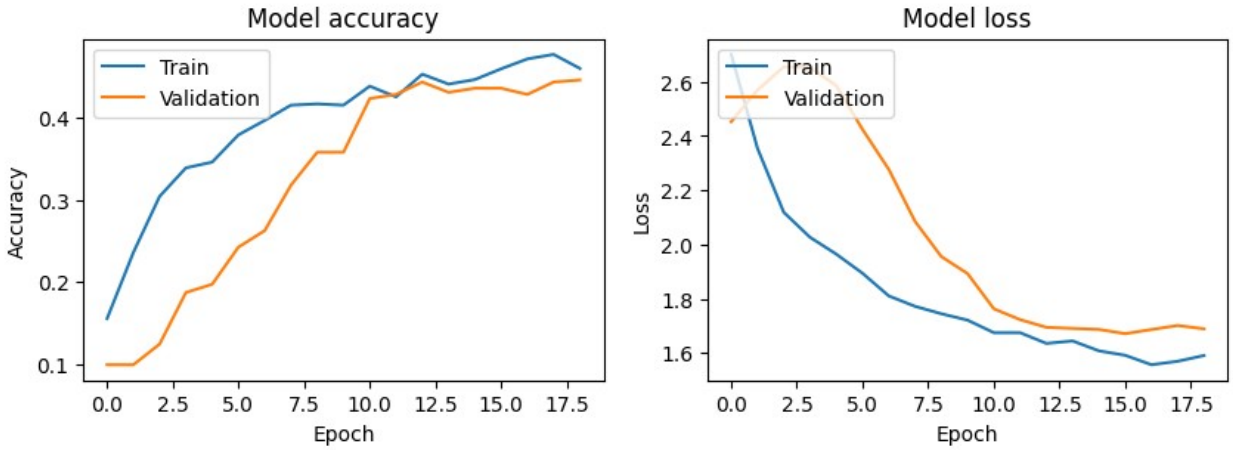
Total training time: 403.90 seconds Test accuracy: 0.5250 Test loss: 1.3801

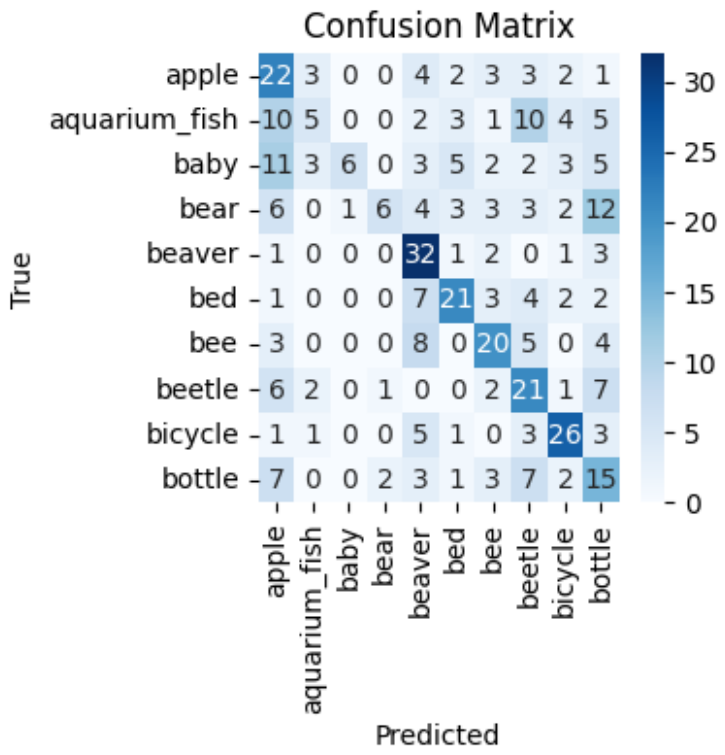
```
cnn4(optimizer_name='sgd', decay_rate=0.85)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
```

```
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 283ms/step - accuracy: 0.1313 - loss:
2.8259 - val_accuracy: 0.1000 - val_loss: 2.4526
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 259ms/step - accuracy: 0.2246 - loss:
2.3338 - val_accuracy: 0.1000 - val_loss: 2.5688
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 282ms/step - accuracy: 0.3030 - loss:
2.1228 - val_accuracy: 0.1250 - val_loss: 2.6547
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 257ms/step - accuracy: 0.3386 - loss:
2.0050 - val_accuracy: 0.1875 - val_loss: 2.6537
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 266ms/step - accuracy: 0.3508 - loss:
1.9400 - val_accuracy: 0.1975 - val_loss: 2.5853
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 277ms/step - accuracy: 0.3610 - loss:
1.9124 - val_accuracy: 0.2425 - val_loss: 2.4239
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 266ms/step - accuracy: 0.3857 - loss:
1.7844 - val_accuracy: 0.2625 - val_loss: 2.2766
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 262ms/step - accuracy: 0.4075 - loss:
1.8196 - val_accuracy: 0.3175 - val_loss: 2.0847
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 268ms/step - accuracy: 0.4138 - loss:
1.7746 - val_accuracy: 0.3575 - val_loss: 1.9551
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 265ms/step - accuracy: 0.4237 - loss:
1.7383 - val_accuracy: 0.3575 - val_loss: 1.8923
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 303ms/step - accuracy: 0.4395 - loss:
1.6600 - val_accuracy: 0.4225 - val_loss: 1.7626
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 269ms/step - accuracy: 0.4184 - loss:
1.7008 - val_accuracy: 0.4275 - val_loss: 1.7226
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 265ms/step - accuracy: 0.4440 - loss:
1.6281 - val_accuracy: 0.4425 - val_loss: 1.6943
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 268ms/step - accuracy: 0.4376 - loss:
1.6345 - val_accuracy: 0.4300 - val_loss: 1.6904
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 266ms/step - accuracy: 0.4620 - loss:
1.5923 - val_accuracy: 0.4350 - val_loss: 1.6866
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 275ms/step - accuracy: 0.4481 - loss:
```

```
1.6105 - val_accuracy: 0.4350 - val_loss: 1.6709
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 257ms/step - accuracy: 0.4789 - loss:
1.5574 - val_accuracy: 0.4275 - val_loss: 1.6862
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 273ms/step - accuracy: 0.4528 - loss:
1.5865 - val_accuracy: 0.4425 - val_loss: 1.7011
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 265ms/step - accuracy: 0.4547 - loss:
1.5943 - val_accuracy: 0.4450 - val_loss: 1.6888
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 57ms/step - accuracy: 0.3811 - loss:
1.8105
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step
Total training time: 380.90 seconds
Test accuracy: 0.4350
Test loss: 1.6709
```






```
<keras.src.callbacks.history.History at 0x7beff8dee890>
```

Total training time: 380.90 seconds Test accuracy: 0.4350 Test loss: 1.6709

SGD doesn't work.

Fifth iteration of the model

```
# Global average pooling instead of flatten

def cnn5(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):
```

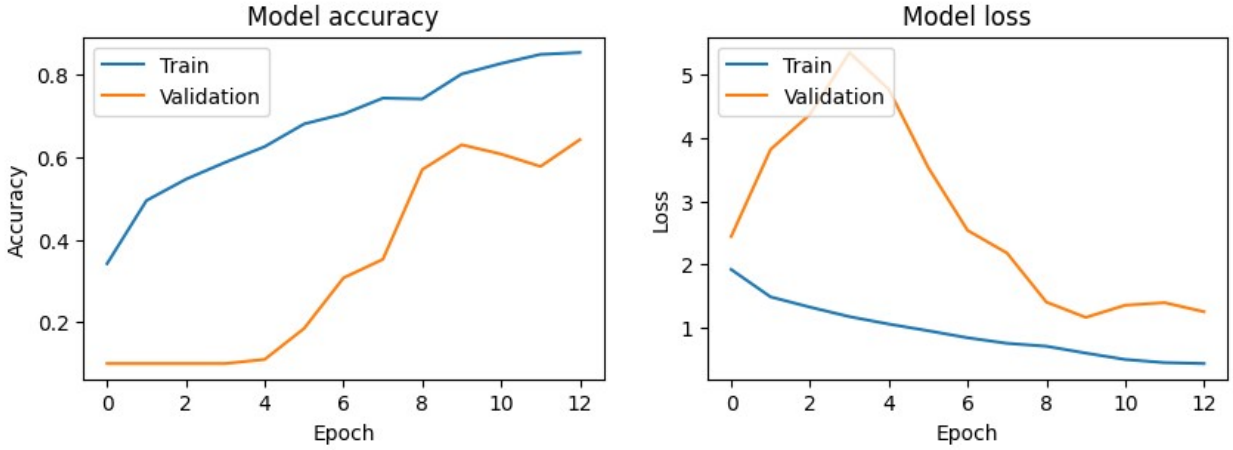
```
    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Global average pooling and Dense layers
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)


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

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
```

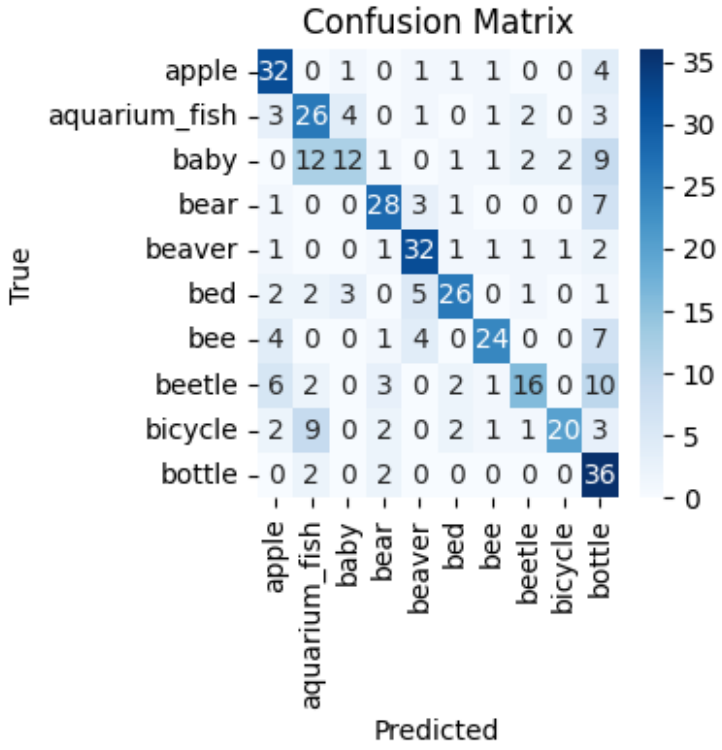
```
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
```

```
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn5()
cnn5(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 33s 356ms/step - accuracy: 0.2758 - loss:
2.1825 - val_accuracy: 0.1000 - val_loss: 2.4428
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 322ms/step - accuracy: 0.4835 - loss:
1.5163 - val_accuracy: 0.1000 - val_loss: 3.8164
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 321ms/step - accuracy: 0.5584 - loss:
1.3196 - val_accuracy: 0.1000 - val_loss: 4.3680
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 272ms/step - accuracy: 0.5853 - loss:
1.1849 - val_accuracy: 0.1000 - val_loss: 5.3445
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 305ms/step - accuracy: 0.6316 - loss:
1.0568 - val_accuracy: 0.1100 - val_loss: 4.7639
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 270ms/step - accuracy: 0.6738 - loss:
0.9756 - val_accuracy: 0.1850 - val_loss: 3.5312
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 269ms/step - accuracy: 0.7059 - loss:
0.8341 - val_accuracy: 0.3075 - val_loss: 2.5385
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 300ms/step - accuracy: 0.7510 - loss:
0.7560 - val_accuracy: 0.3525 - val_loss: 2.1809
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 261ms/step - accuracy: 0.7517 - loss:
0.6881 - val_accuracy: 0.5700 - val_loss: 1.4071
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 281ms/step - accuracy: 0.8097 - loss:
0.5833 - val_accuracy: 0.6300 - val_loss: 1.1656
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 261ms/step - accuracy: 0.8217 - loss:
0.5061 - val_accuracy: 0.6075 - val_loss: 1.3578
Epoch 12/20
```

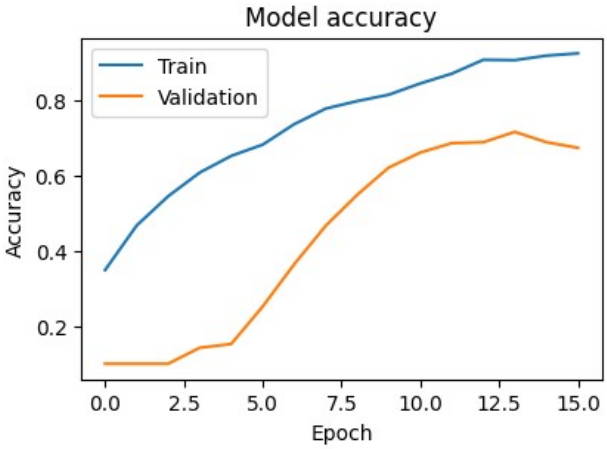
```
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 276ms/step - accuracy: 0.8419 - loss:
0.4634 - val_accuracy: 0.5775 - val_loss: 1.3979
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 263ms/step - accuracy: 0.8587 - loss:
0.4212 - val_accuracy: 0.6425 - val_loss: 1.2554
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 102ms/step - accuracy: 0.6480 - loss:
1.1367
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 74ms/step
Total training time: 271.04 seconds
Test accuracy: 0.6300
Test loss: 1.1656
```

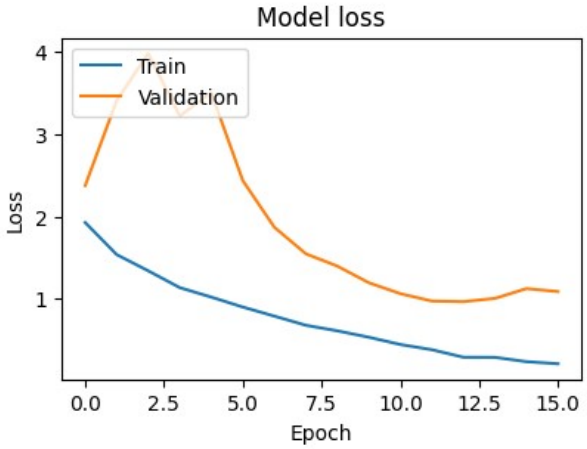


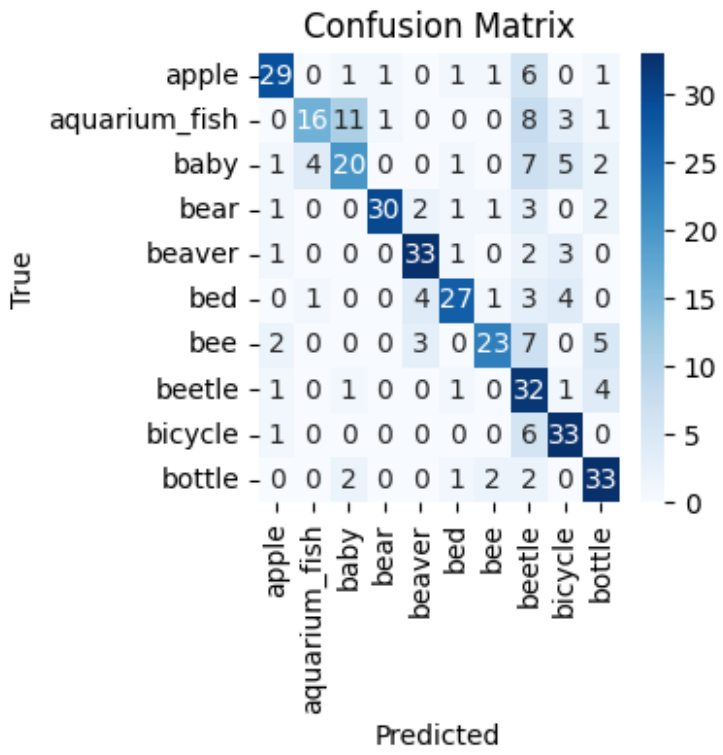


```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 30s 363ms/step - accuracy: 0.3085 - loss:
2.1720 - val_accuracy: 0.1000 - val_loss: 2.3773
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 25s 406ms/step - accuracy: 0.4439 - loss:
1.5912 - val_accuracy: 0.1000 - val_loss: 3.4123
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 34s 299ms/step - accuracy: 0.5410 - loss:
1.3721 - val_accuracy: 0.1000 - val_loss: 3.9843
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 290ms/step - accuracy: 0.6077 - loss:
1.1333 - val_accuracy: 0.1425 - val_loss: 3.2247
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 308ms/step - accuracy: 0.6610 - loss:
0.9944 - val_accuracy: 0.1525 - val_loss: 3.5047
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 286ms/step - accuracy: 0.6928 - loss:
0.9150 - val_accuracy: 0.2525 - val_loss: 2.4381
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 310ms/step - accuracy: 0.7530 - loss:
0.7722 - val_accuracy: 0.3650 - val_loss: 1.8712
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 315ms/step - accuracy: 0.7918 - loss:
0.6323 - val_accuracy: 0.4675 - val_loss: 1.5482
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 287ms/step - accuracy: 0.8109 - loss:
0.6092 - val_accuracy: 0.5500 - val_loss: 1.3983
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 315ms/step - accuracy: 0.8264 - loss:
0.5080 - val_accuracy: 0.6225 - val_loss: 1.1960
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 278ms/step - accuracy: 0.8562 - loss:
0.4233 - val_accuracy: 0.6625 - val_loss: 1.0633
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 287ms/step - accuracy: 0.8640 - loss:
0.3920 - val_accuracy: 0.6875 - val_loss: 0.9736
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 288ms/step - accuracy: 0.9065 - loss:
0.2869 - val_accuracy: 0.6900 - val_loss: 0.9665
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 298ms/step - accuracy: 0.9257 - loss:
0.2550 - val_accuracy: 0.7175 - val_loss: 1.0061
```

```
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 279ms/step - accuracy: 0.9222 - loss:
0.2372 - val_accuracy: 0.6900 - val_loss: 1.1249
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 299ms/step - accuracy: 0.9254 - loss:
0.2175 - val_accuracy: 0.6750 - val_loss: 1.0891
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step - accuracy: 0.6408 - loss:
1.0965
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step
Total training time: 345.35 seconds
Test accuracy: 0.6900
Test loss: 0.9665
```








```
<keras.src.callbacks.history.History at 0x7befd579c910>
```

Total training time: 271.04 seconds Test accuracy: 0.6300 Test loss: 1.1656

Base version of this model isn't any better

Total training time: 345.35 seconds Test accuracy: 0.6900 Test loss: 0.9665

With swish it performs pretty close to the best one yet.

##Sixth iteration of the model

```
# Adding another dense layer

def cnn6(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
```

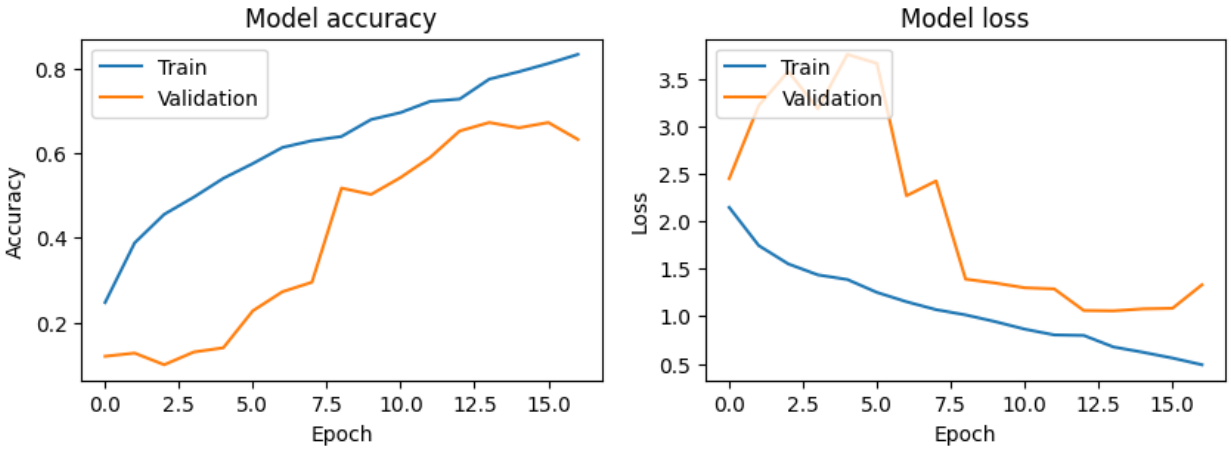
```
        batch_size=32,
        ):

    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Global average pooling and Dense layers
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(dense_units // 2, activation=activation))
    model.add(layers.Dropout(dropout))
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)


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

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True,
```

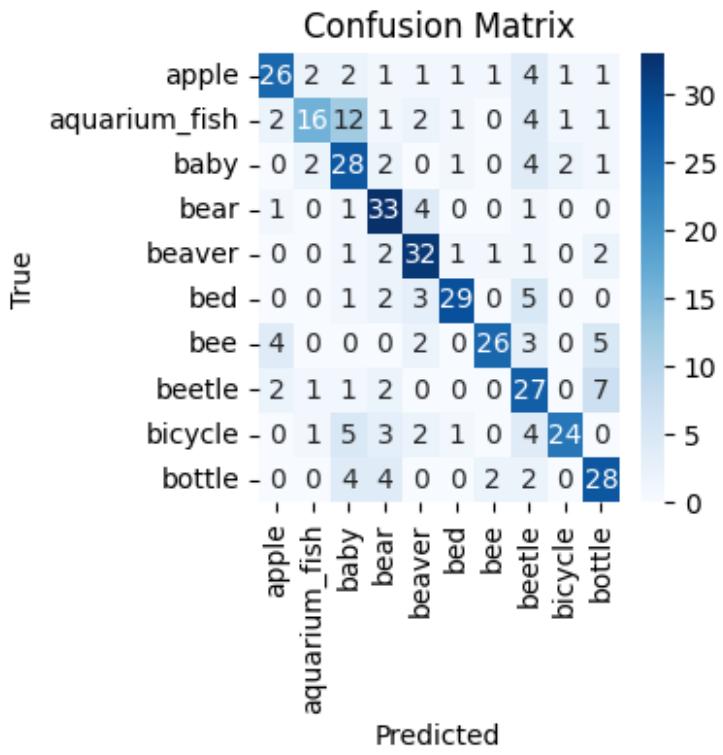
```
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
```

```
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn6()
cnn6(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 294ms/step - accuracy: 0.1914 - loss:
2.3435 - val_accuracy: 0.1200 - val_loss: 2.4458
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 16s 258ms/step - accuracy: 0.3662 - loss:
1.7945 - val_accuracy: 0.1275 - val_loss: 3.2152
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 278ms/step - accuracy: 0.4506 - loss:
1.5582 - val_accuracy: 0.1000 - val_loss: 3.5740
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 260ms/step - accuracy: 0.4964 - loss:
1.4234 - val_accuracy: 0.1300 - val_loss: 3.1822
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 279ms/step - accuracy: 0.5550 - loss:
1.3645 - val_accuracy: 0.1400 - val_loss: 3.7530
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 271ms/step - accuracy: 0.5745 - loss:
1.2120 - val_accuracy: 0.2275 - val_loss: 3.6580
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 270ms/step - accuracy: 0.6152 - loss:
1.1607 - val_accuracy: 0.2725 - val_loss: 2.2666
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 270ms/step - accuracy: 0.6186 - loss:
1.0810 - val_accuracy: 0.2950 - val_loss: 2.4216
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.6310 - loss:
1.0260 - val_accuracy: 0.5175 - val_loss: 1.3893
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 258ms/step - accuracy: 0.6631 - loss:
```

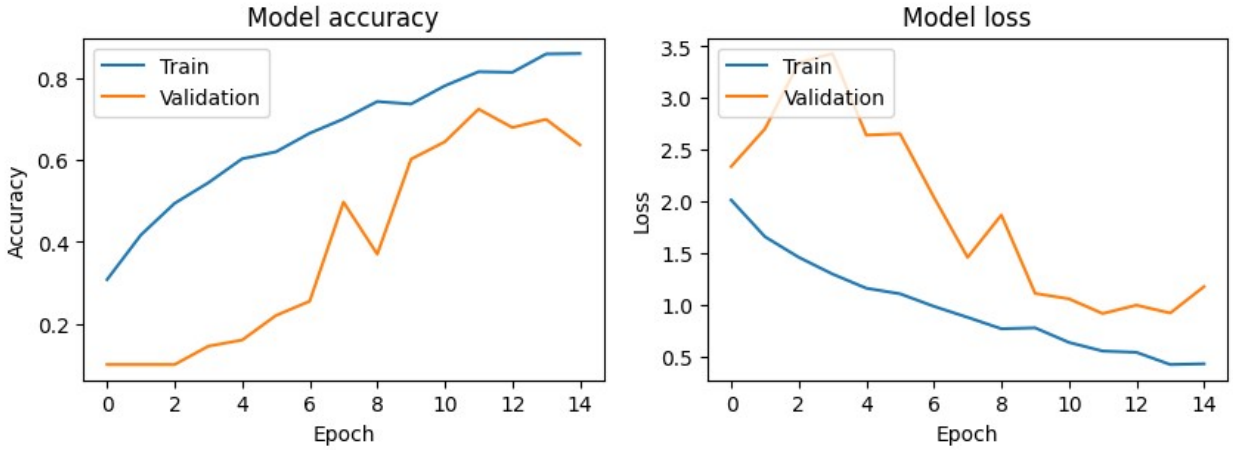
```
0.9579 - val_accuracy: 0.5025 - val_loss: 1.3488
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 277ms/step - accuracy: 0.7033 - loss:
0.8595 - val_accuracy: 0.5425 - val_loss: 1.2987
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 269ms/step - accuracy: 0.7269 - loss:
0.7881 - val_accuracy: 0.5900 - val_loss: 1.2869
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 272ms/step - accuracy: 0.7379 - loss:
0.7890 - val_accuracy: 0.6525 - val_loss: 1.0598
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 290ms/step - accuracy: 0.7667 - loss:
0.6870 - val_accuracy: 0.6725 - val_loss: 1.0560
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 268ms/step - accuracy: 0.8160 - loss:
0.5708 - val_accuracy: 0.6600 - val_loss: 1.0769
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 269ms/step - accuracy: 0.7975 - loss:
0.5900 - val_accuracy: 0.6725 - val_loss: 1.0826
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 24s 323ms/step - accuracy: 0.8418 - loss:
0.4681 - val_accuracy: 0.6325 - val_loss: 1.3301
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 60ms/step - accuracy: 0.6451 - loss:
1.1383
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 75ms/step
Total training time: 335.84 seconds
Test accuracy: 0.6725
Test loss: 1.0560
```

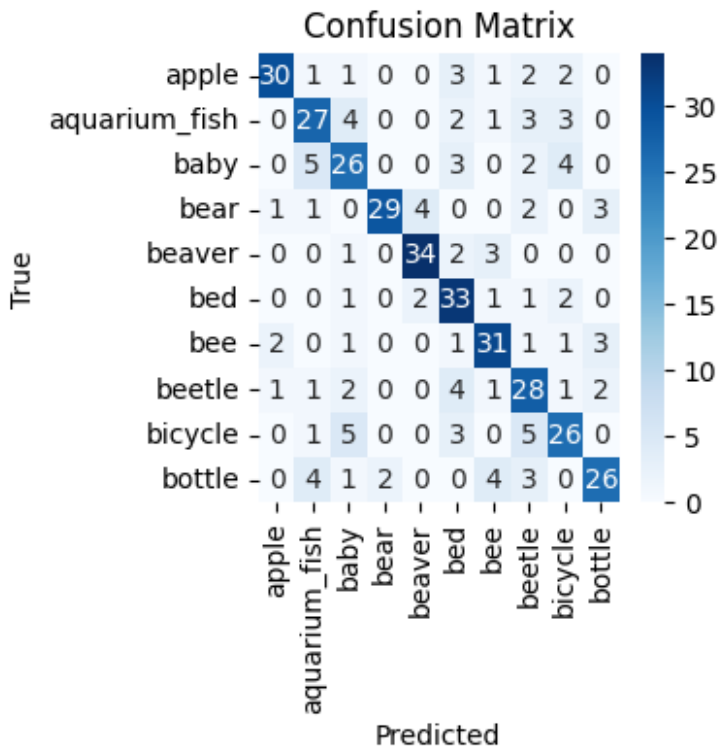





```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 32s 366ms/step - accuracy: 0.2542 - loss:
2.1600 - val_accuracy: 0.1000 - val_loss: 2.3375
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 335ms/step - accuracy: 0.4186 - loss:
1.6579 - val_accuracy: 0.1000 - val_loss: 2.7025
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 298ms/step - accuracy: 0.4897 - loss:
1.4589 - val_accuracy: 0.1000 - val_loss: 3.3413
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 288ms/step - accuracy: 0.5451 - loss:
1.2984 - val_accuracy: 0.1450 - val_loss: 3.4314
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 313ms/step - accuracy: 0.6130 - loss:
1.1476 - val_accuracy: 0.1600 - val_loss: 2.6418
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 413ms/step - accuracy: 0.6244 - loss:
1.1006 - val_accuracy: 0.2200 - val_loss: 2.6547
```

```
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 35s 315ms/step - accuracy: 0.6724 - loss:
0.9782 - val_accuracy: 0.2550 - val_loss: 2.0417
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 286ms/step - accuracy: 0.6916 - loss:
0.9110 - val_accuracy: 0.4975 - val_loss: 1.4578
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 28s 446ms/step - accuracy: 0.7491 - loss:
0.7445 - val_accuracy: 0.3700 - val_loss: 1.8683
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 32s 301ms/step - accuracy: 0.7461 - loss:
0.7923 - val_accuracy: 0.6025 - val_loss: 1.1090
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 290ms/step - accuracy: 0.7840 - loss:
0.6267 - val_accuracy: 0.6450 - val_loss: 1.0571
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 306ms/step - accuracy: 0.8214 - loss:
0.5369 - val_accuracy: 0.7250 - val_loss: 0.9143
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 297ms/step - accuracy: 0.8025 - loss:
0.5598 - val_accuracy: 0.6800 - val_loss: 0.9954
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 307ms/step - accuracy: 0.8651 - loss:
0.4075 - val_accuracy: 0.7000 - val_loss: 0.9191
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 288ms/step - accuracy: 0.8710 - loss:
0.4118 - val_accuracy: 0.6375 - val_loss: 1.1749
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step - accuracy: 0.7248 - loss:
0.9560
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 86ms/step
Total training time: 373.63 seconds
Test accuracy: 0.7250
Test loss: 0.9143
```






```
<keras.src.callbacks.history.History at 0x7befd79032d0>
```

Total training time: 335.84 seconds Test accuracy: 0.6725 Test loss: 1.0560

This one is pretty close to the original baseline, but doesnt seem to be overfitting that badly.

Total training time: 373.63 seconds Test accuracy: 0.7250 Test loss: 0.9143

These are the best scores so far. And the model isn't overfitting so bad after applying the second dense layer.

(I just now noticed that the class names in the confusion matrices are wrong. I corrected my code in the beginning of the notebook so they should be correct from this point on)

Seventh iteration of the model

```
# Switching the loop a bit, now BatchNormalization is before
activation layer.

def cnn7(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
```

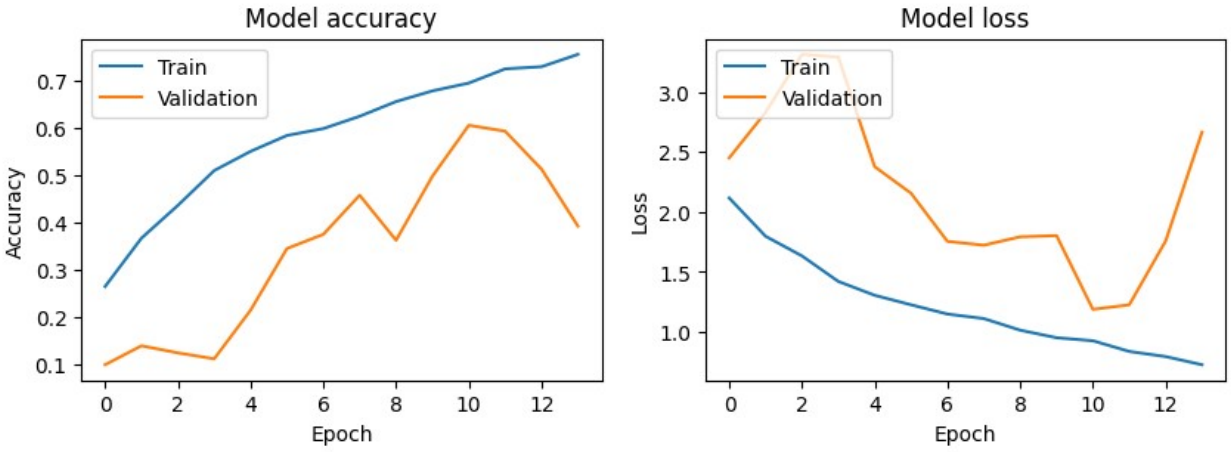
```
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):

    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
use_bias=False,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Activation(activation))

        model.add(layers.Conv2D(filters, (3, 3), padding='same',
use_bias=False))
        model.add(layers.BatchNormalization())
        model.add(layers.Activation(activation))

        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Global average pooling and Dense layers
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(dense_units // 2, activation=activation))
    model.add(layers.Dropout(dropout))
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)
```

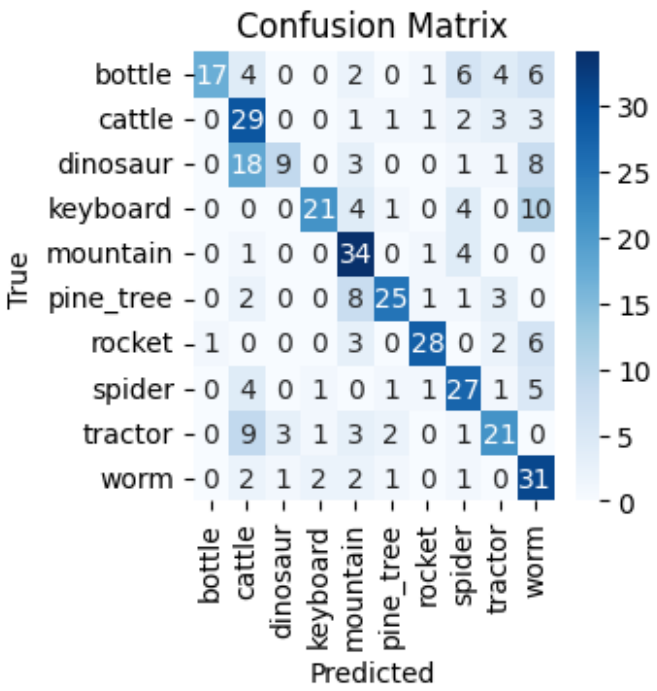
```
    # Compile
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
```

```
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn7()
cnn7(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 25s 285ms/step - accuracy: 0.2220 - loss:
2.2686 - val_accuracy: 0.1000 - val_loss: 2.4504
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 274ms/step - accuracy: 0.3576 - loss:
1.8305 - val_accuracy: 0.1400 - val_loss: 2.8305
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 270ms/step - accuracy: 0.4284 - loss:
1.6432 - val_accuracy: 0.1250 - val_loss: 3.3136
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 272ms/step - accuracy: 0.5420 - loss:
1.3960 - val_accuracy: 0.1125 - val_loss: 3.2916
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 282ms/step - accuracy: 0.5476 - loss:
1.3278 - val_accuracy: 0.2150 - val_loss: 2.3775
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 272ms/step - accuracy: 0.5810 - loss:
1.2174 - val_accuracy: 0.3450 - val_loss: 2.1560
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 293ms/step - accuracy: 0.6085 - loss:
```

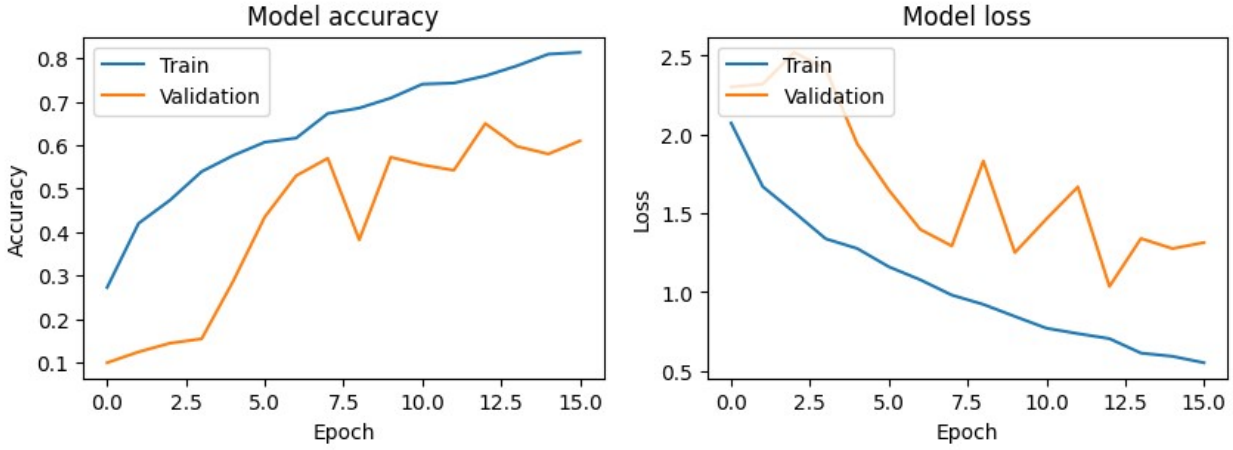
```
1.0987 - val_accuracy: 0.3750 - val_loss: 1.7548
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 270ms/step - accuracy: 0.6410 - loss:
1.0625 - val_accuracy: 0.4575 - val_loss: 1.7229
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 262ms/step - accuracy: 0.6559 - loss:
1.0208 - val_accuracy: 0.3625 - val_loss: 1.7929
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 272ms/step - accuracy: 0.6738 - loss:
0.9521 - val_accuracy: 0.4975 - val_loss: 1.8029
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 272ms/step - accuracy: 0.6891 - loss:
0.9722 - val_accuracy: 0.6050 - val_loss: 1.1881
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 280ms/step - accuracy: 0.7383 - loss:
0.8154 - val_accuracy: 0.5925 - val_loss: 1.2258
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 263ms/step - accuracy: 0.7301 - loss:
0.7702 - val_accuracy: 0.5125 - val_loss: 1.7576
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 273ms/step - accuracy: 0.7732 - loss:
0.6754 - val_accuracy: 0.3925 - val_loss: 2.6648
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 58ms/step - accuracy: 0.5514 - loss:
1.3650
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step
Total training time: 279.96 seconds
Test accuracy: 0.6050
Test loss: 1.1881
```

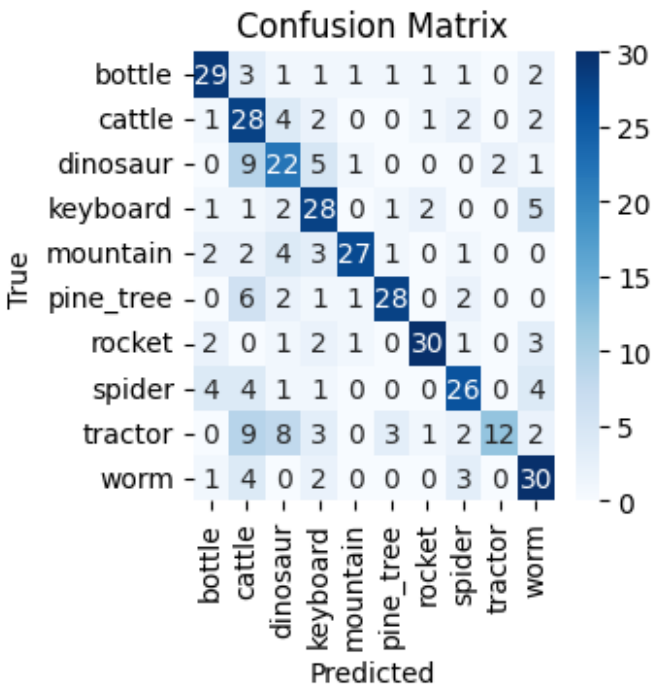





```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 26s 286ms/step - accuracy: 0.2119 - loss:
2.2607 - val_accuracy: 0.1000 - val_loss: 2.3008
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 297ms/step - accuracy: 0.4219 - loss:
1.6850 - val_accuracy: 0.1250 - val_loss: 2.3186
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 287ms/step - accuracy: 0.4696 - loss:
1.5294 - val_accuracy: 0.1450 - val_loss: 2.5203
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 305ms/step - accuracy: 0.5398 - loss:
1.3368 - val_accuracy: 0.1550 - val_loss: 2.4156
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 280ms/step - accuracy: 0.5821 - loss:
1.2607 - val_accuracy: 0.2875 - val_loss: 1.9394
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 302ms/step - accuracy: 0.6143 - loss:
1.1450 - val_accuracy: 0.4350 - val_loss: 1.6471
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 294ms/step - accuracy: 0.5945 - loss:
```

```
1.1242 - val_accuracy: 0.5300 - val_loss: 1.3975
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 286ms/step - accuracy: 0.6631 - loss:
0.9922 - val_accuracy: 0.5700 - val_loss: 1.2927
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 305ms/step - accuracy: 0.6775 - loss:
0.9429 - val_accuracy: 0.3825 - val_loss: 1.8318
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 350ms/step - accuracy: 0.7077 - loss:
0.8366 - val_accuracy: 0.5725 - val_loss: 1.2507
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 38s 306ms/step - accuracy: 0.7554 - loss:
0.7482 - val_accuracy: 0.5550 - val_loss: 1.4639
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 287ms/step - accuracy: 0.7536 - loss:
0.6916 - val_accuracy: 0.5425 - val_loss: 1.6674
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 303ms/step - accuracy: 0.7648 - loss:
0.6816 - val_accuracy: 0.6500 - val_loss: 1.0356
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 287ms/step - accuracy: 0.7906 - loss:
0.5977 - val_accuracy: 0.5975 - val_loss: 1.3399
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 306ms/step - accuracy: 0.8208 - loss:
0.5394 - val_accuracy: 0.5800 - val_loss: 1.2760
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 281ms/step - accuracy: 0.8265 - loss:
0.5207 - val_accuracy: 0.6100 - val_loss: 1.3141
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 82ms/step - accuracy: 0.6730 - loss:
0.9959
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 119ms/step
Total training time: 348.76 seconds
Test accuracy: 0.6500
Test loss: 1.0356
```






```
<keras.src.callbacks.history.History at 0x7befcfae7a50>
```

Total training time: 279.96 seconds Test accuracy: 0.6050 Test loss: 1.1881

Total training time: 348.76 seconds Test accuracy: 0.6500 Test loss: 1.0356

This didn't help at all.

Eight iteration of the model

```
# Switching back to flatten and previous loop config.

def cnn8(train_data=x_train_selected,
        train_labels=y_train_selected,
        val_data=x_test_selected,
        val_labels=y_test_selected,
        activation='relu',
        dropout=0.2,
        optimizer_name='adam',
        learning_rate=0.001,
        decay_rate=0.9,
        epochs=20,
        conv_layers=3,
        filters = 32,
        dense_units=128,
        batch_size=32,
        ):
```

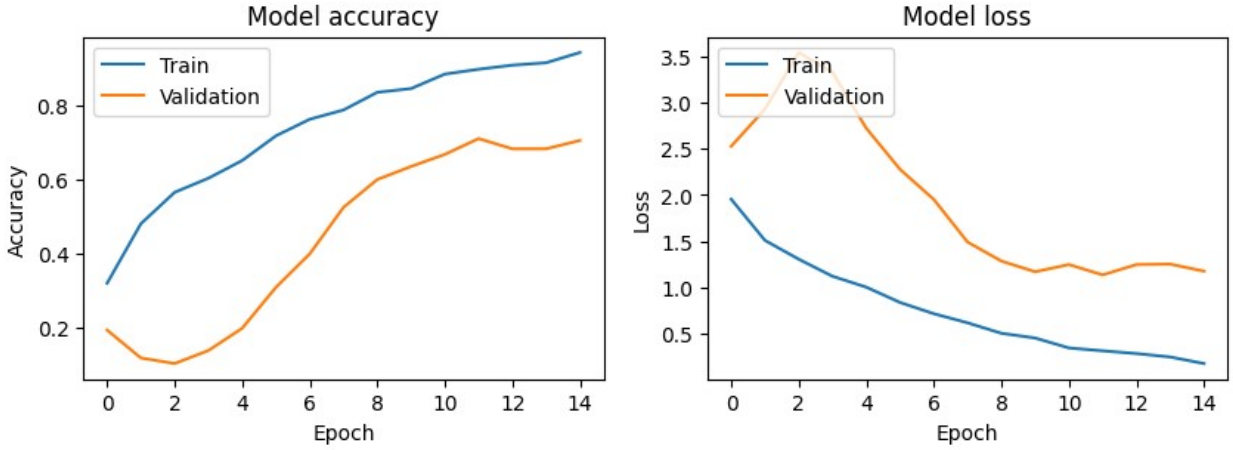
```
    # Initialize model
    model = Sequential()

    # Loop for Convolutional layers
    for i in range(conv_layers):
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation,
                                input_shape=(32, 32, 3) if i == 0 else
None))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(filters, (3, 3), padding='same',
activation=activation))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(dropout))
        filters *= 2

    # Flatten and Dense layers
    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation=activation))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(dense_units // 2, activation=activation))
    model.add(layers.Dropout(dropout))
    model.add(layers.Dense(len(classes), activation='softmax'))

    # Optimizer
    optimizer = optimizers.get(optimizer_name)
    if optimizer_name == 'adam' or optimizer_name == 'rmsprop':
        optimizer.learning_rate.assign(learning_rate)
    elif optimizer_name == 'sgd':
        lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
            initial_learning_rate=learning_rate,
            decay_steps=100,
            decay_rate=decay_rate
        )
        optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule)


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

    # Early stopping
    early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    start_from_epoch=5)

    # Training
```

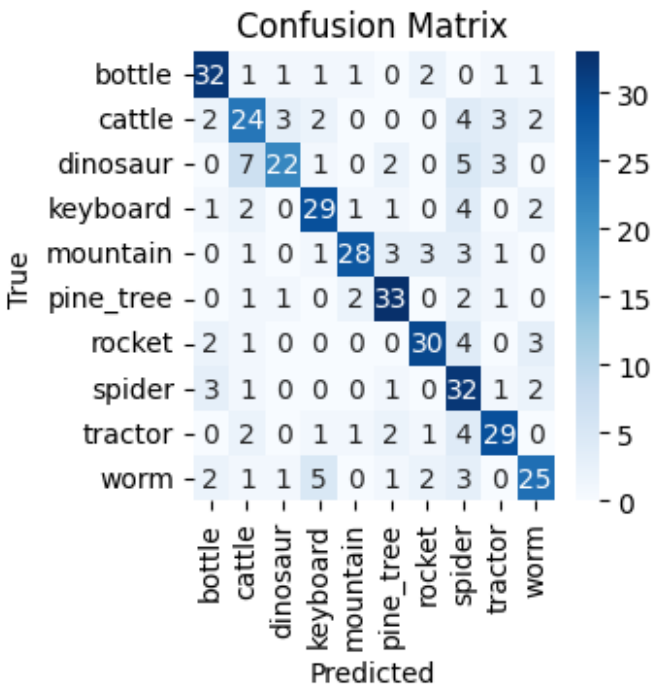
```
    start_time = time.time()  # Start timer
    history = model.fit(train_data,
                        train_labels,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(val_data, val_labels),
                        verbose=1,
                        callbacks=[early_stopping])


    end_time = time.time()  # End timer

    # Model evaluation
    test_loss, test_acc = model.evaluate(val_data, val_labels)

    y_pred = model.predict(val_data)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(val_labels, axis=1)

    # Print testing accuracy and loss
    print(f"Total training time: {end_time - start_time:.2f} seconds")
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

    # Plot training and validation accuracy and loss
    fig, axes = plt.subplots(1, 2, figsize=(10, 3))

    axes[0].plot(history.history['accuracy'])
    axes[0].plot(history.history['val_accuracy'])
    axes[0].set_title('Model accuracy')
    axes[0].set_ylabel('Accuracy')
    axes[0].set_xlabel('Epoch')
    axes[0].legend(['Train', 'Validation'], loc='upper left')

    axes[1].plot(history.history['loss'])
    axes[1].plot(history.history['val_loss'])
    axes[1].set_title('Model loss')
    axes[1].set_ylabel('Loss')
    axes[1].set_xlabel('Epoch')
    axes[1].legend(['Train', 'Validation'], loc='upper left')

    plt.show()

    # Plot confusion matrix
    cm = confusion_matrix(y_true_classes, y_pred_classes)

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=selected_class_names,
                yticklabels=selected_class_names)
```

```
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return history

cnn8(activation='swish')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 28s 331ms/step - accuracy: 0.2544 - loss:
2.1885 - val_accuracy: 0.1950 - val_loss: 2.5277
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 291ms/step - accuracy: 0.4860 - loss:
1.5043 - val_accuracy: 0.1200 - val_loss: 2.9288
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 305ms/step - accuracy: 0.5574 - loss:
1.3372 - val_accuracy: 0.1050 - val_loss: 3.5408
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 290ms/step - accuracy: 0.6032 - loss:
1.1385 - val_accuracy: 0.1400 - val_loss: 3.3280
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 324ms/step - accuracy: 0.6551 - loss:
1.0070 - val_accuracy: 0.2000 - val_loss: 2.7225
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 293ms/step - accuracy: 0.7251 - loss:
0.8079 - val_accuracy: 0.3100 - val_loss: 2.2785
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 288ms/step - accuracy: 0.7790 - loss:
0.6691 - val_accuracy: 0.4000 - val_loss: 1.9529
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 305ms/step - accuracy: 0.8100 - loss:
0.5771 - val_accuracy: 0.5250 - val_loss: 1.4923
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 293ms/step - accuracy: 0.8482 - loss:
0.4781 - val_accuracy: 0.6000 - val_loss: 1.2872
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 310ms/step - accuracy: 0.8487 - loss:
0.4500 - val_accuracy: 0.6350 - val_loss: 1.1717
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 19s 288ms/step - accuracy: 0.9071 - loss:
0.3153 - val_accuracy: 0.6675 - val_loss: 1.2494
```

```
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 312ms/step - accuracy: 0.9017 - loss:
0.3095 - val_accuracy: 0.7100 - val_loss: 1.1375
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 18s 285ms/step - accuracy: 0.9088 - loss:
0.2783 - val_accuracy: 0.6825 - val_loss: 1.2494
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 300ms/step - accuracy: 0.9119 - loss:
0.2597 - val_accuracy: 0.6825 - val_loss: 1.2541
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 316ms/step - accuracy: 0.9461 - loss:
0.1801 - val_accuracy: 0.7050 - val_loss: 1.1781
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 64ms/step - accuracy: 0.7066 - loss:
1.2458
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 84ms/step
Total training time: 331.05 seconds
Test accuracy: 0.7100
Test loss: 1.1375
```






```
<keras.src.callbacks.history.History at 0x7befd7a66410>
```

Total training time: 331.05 seconds Test accuracy: 0.7100 Test loss: 1.1375

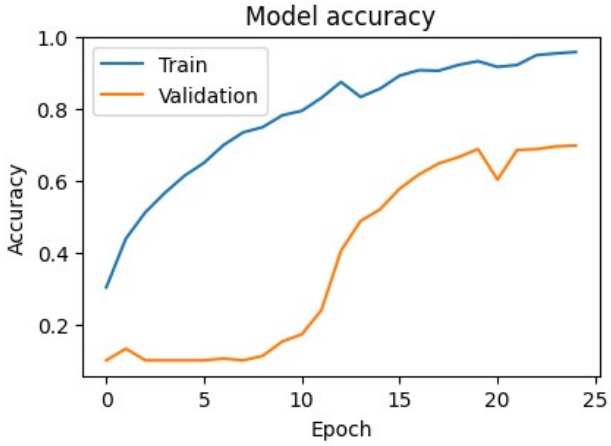
This was closer to iter6 but not as good. I will stick with iter6 and experiment a little bit more with it.

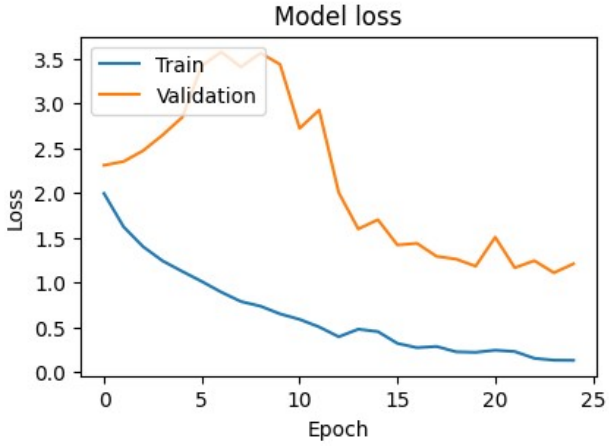
Tuning the best model configuration

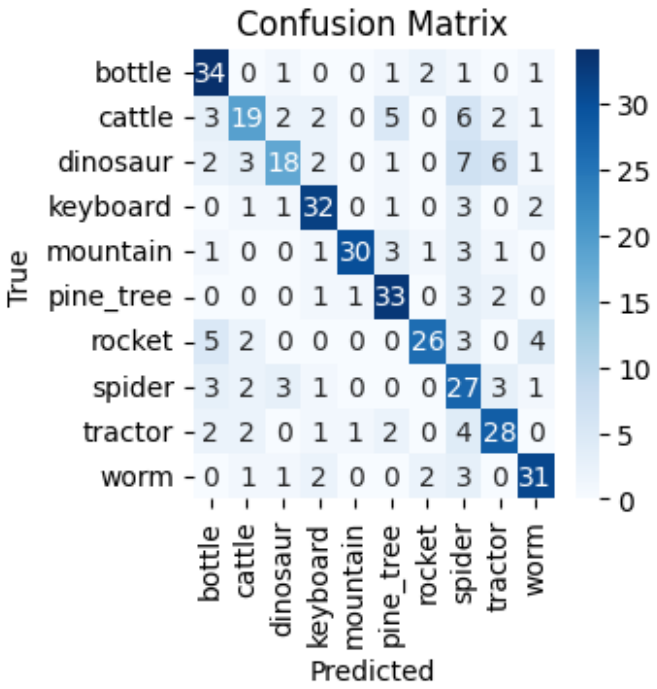
```
cnn6(activation='swish', dense_units=256, batch_size=64, epochs=25)

/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 30s 673ms/step - accuracy: 0.2435 - loss:
2.1444 - val_accuracy: 0.1000 - val_loss: 2.3070
Epoch 2/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 41s 674ms/step - accuracy: 0.4009 - loss:
1.7354 - val_accuracy: 0.1325 - val_loss: 2.3485
Epoch 3/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 40s 648ms/step - accuracy: 0.5137 - loss:
```

```
1.3964 - val_accuracy: 0.1000 - val_loss: 2.4696
Epoch 4/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 619ms/step - accuracy: 0.5508 - loss:
1.2573 - val_accuracy: 0.1000 - val_loss: 2.6447
Epoch 5/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 653ms/step - accuracy: 0.6104 - loss:
1.1221 - val_accuracy: 0.1000 - val_loss: 2.8421
Epoch 6/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 644ms/step - accuracy: 0.6516 - loss:
0.9823 - val_accuracy: 0.1000 - val_loss: 3.4193
Epoch 7/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 42s 687ms/step - accuracy: 0.6930 - loss:
0.8984 - val_accuracy: 0.1050 - val_loss: 3.5707
Epoch 8/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 40s 636ms/step - accuracy: 0.7431 - loss:
0.7583 - val_accuracy: 0.1000 - val_loss: 3.4020
Epoch 9/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 671ms/step - accuracy: 0.7683 - loss:
0.6952 - val_accuracy: 0.1125 - val_loss: 3.5594
Epoch 10/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 41s 669ms/step - accuracy: 0.7781 - loss:
0.6451 - val_accuracy: 0.1525 - val_loss: 3.4323
Epoch 11/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 41s 655ms/step - accuracy: 0.7911 - loss:
0.6013 - val_accuracy: 0.1725 - val_loss: 2.7189
Epoch 12/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 40s 631ms/step - accuracy: 0.8404 - loss:
0.4788 - val_accuracy: 0.2400 - val_loss: 2.9227
Epoch 13/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 651ms/step - accuracy: 0.8742 - loss:
0.3952 - val_accuracy: 0.4050 - val_loss: 2.0012
Epoch 14/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 645ms/step - accuracy: 0.8559 - loss:
0.4130 - val_accuracy: 0.4875 - val_loss: 1.5957
Epoch 15/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 41s 650ms/step - accuracy: 0.8595 - loss:
0.4642 - val_accuracy: 0.5200 - val_loss: 1.6998
Epoch 16/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 40s 622ms/step - accuracy: 0.8990 - loss:
0.2984 - val_accuracy: 0.5775 - val_loss: 1.4177
Epoch 17/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 638ms/step - accuracy: 0.9152 - loss:
0.2507 - val_accuracy: 0.6175 - val_loss: 1.4353
Epoch 18/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 609ms/step - accuracy: 0.9157 - loss:
0.2454 - val_accuracy: 0.6475 - val_loss: 1.2908
Epoch 19/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 670ms/step - accuracy: 0.9237 - loss:
0.2408 - val_accuracy: 0.6650 - val_loss: 1.2596
```

```
Epoch 20/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 614ms/step - accuracy: 0.9382 - loss:
0.2017 - val_accuracy: 0.6875 - val_loss: 1.1806
Epoch 21/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 22s 663ms/step - accuracy: 0.9257 - loss:
0.2110 - val_accuracy: 0.6025 - val_loss: 1.5047
Epoch 22/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 40s 631ms/step - accuracy: 0.9237 - loss:
0.2198 - val_accuracy: 0.6850 - val_loss: 1.1632
Epoch 23/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 21s 672ms/step - accuracy: 0.9501 - loss:
0.1531 - val_accuracy: 0.6875 - val_loss: 1.2419
Epoch 24/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 41s 664ms/step - accuracy: 0.9491 - loss:
0.1351 - val_accuracy: 0.6950 - val_loss: 1.1070
Epoch 25/25
32/32 ━━━━━━━━━━━━━━━━━━━━ 20s 634ms/step - accuracy: 0.9618 - loss:
0.1122 - val_accuracy: 0.6975 - val_loss: 1.2071
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 129ms/step - accuracy: 0.6867 - loss:
1.1530
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 99ms/step
Total training time: 748.27 seconds
Test accuracy: 0.6950
Test loss: 1.1070
```







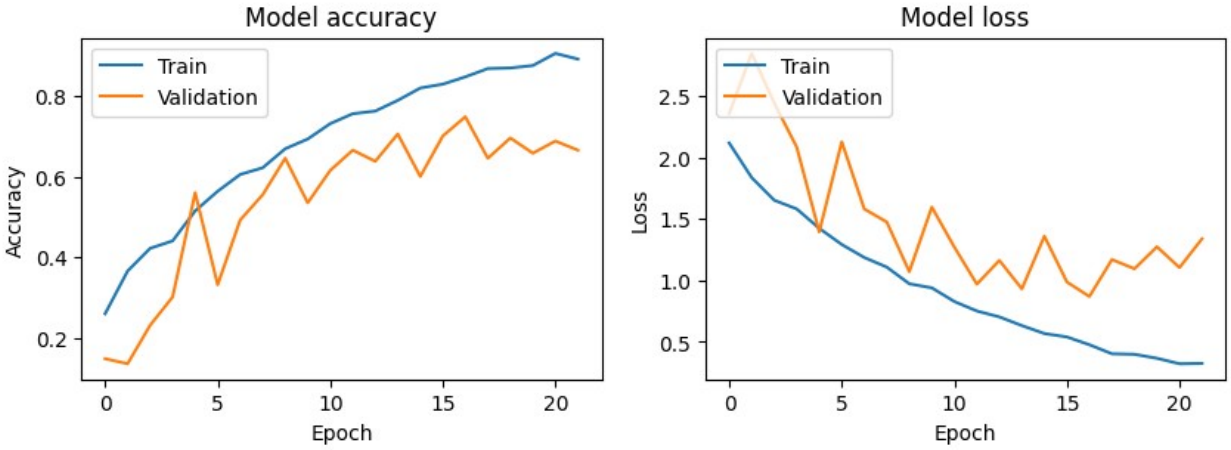

```
<keras.src.callbacks.history.History at 0x780eca677a90>
```

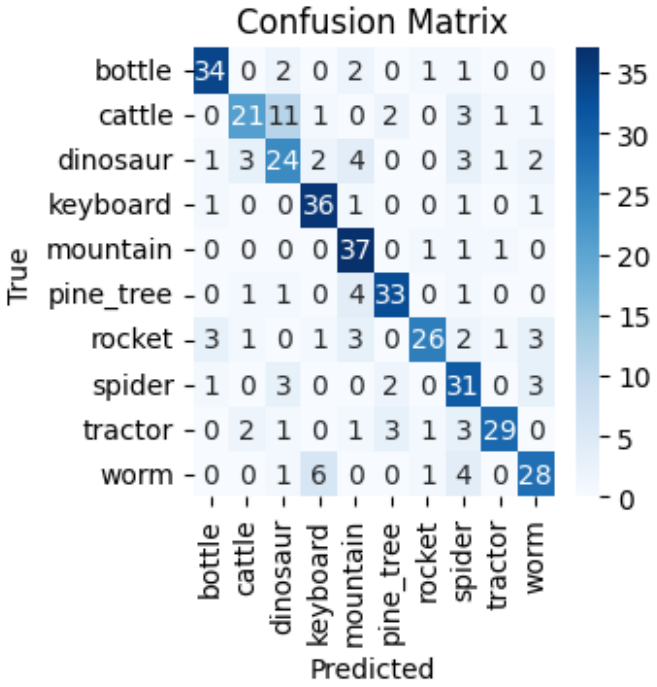
```
cnn6(activation='swish', dense_units=64, batch_size=16, epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 28s 171ms/step - accuracy: 0.2200 - loss:
2.2530 - val_accuracy: 0.1500 - val_loss: 2.3606
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 164ms/step - accuracy: 0.3459 - loss:
1.8722 - val_accuracy: 0.1375 - val_loss: 2.8500
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 158ms/step - accuracy: 0.4304 - loss:
1.6649 - val_accuracy: 0.2325 - val_loss: 2.4452
Epoch 4/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 167ms/step - accuracy: 0.4320 - loss:
1.6255 - val_accuracy: 0.3025 - val_loss: 2.0861
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 169ms/step - accuracy: 0.4940 - loss:
1.4422 - val_accuracy: 0.5600 - val_loss: 1.3960
Epoch 6/25
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.5607 - loss:
1.2840 - val_accuracy: 0.3325 - val_loss: 2.1297
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 157ms/step - accuracy: 0.6011 - loss:
1.1850 - val_accuracy: 0.4925 - val_loss: 1.5826
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 158ms/step - accuracy: 0.6171 - loss:
1.1140 - val_accuracy: 0.5550 - val_loss: 1.4765
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 166ms/step - accuracy: 0.6876 - loss:
0.9375 - val_accuracy: 0.6450 - val_loss: 1.0720
Epoch 10/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.6790 - loss:
0.9633 - val_accuracy: 0.5350 - val_loss: 1.5981
Epoch 11/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 157ms/step - accuracy: 0.7523 - loss:
0.8003 - val_accuracy: 0.6150 - val_loss: 1.2723
Epoch 12/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 164ms/step - accuracy: 0.7588 - loss:
0.7109 - val_accuracy: 0.6650 - val_loss: 0.9707
Epoch 13/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 155ms/step - accuracy: 0.7678 - loss:
0.6950 - val_accuracy: 0.6375 - val_loss: 1.1640
Epoch 14/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 167ms/step - accuracy: 0.7984 - loss:
0.6207 - val_accuracy: 0.7050 - val_loss: 0.9321
Epoch 15/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 157ms/step - accuracy: 0.8286 - loss:
0.5455 - val_accuracy: 0.6000 - val_loss: 1.3613
Epoch 16/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 166ms/step - accuracy: 0.8191 - loss:
0.5614 - val_accuracy: 0.7000 - val_loss: 0.9880
Epoch 17/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.8453 - loss:
0.4785 - val_accuracy: 0.7475 - val_loss: 0.8698
Epoch 18/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 157ms/step - accuracy: 0.8788 - loss:
0.3749 - val_accuracy: 0.6450 - val_loss: 1.1710
Epoch 19/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 166ms/step - accuracy: 0.8734 - loss:
0.3914 - val_accuracy: 0.6950 - val_loss: 1.0957
Epoch 20/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 158ms/step - accuracy: 0.8835 - loss:
0.3376 - val_accuracy: 0.6575 - val_loss: 1.2748
Epoch 21/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 168ms/step - accuracy: 0.9143 - loss:
0.2996 - val_accuracy: 0.6875 - val_loss: 1.1054
Epoch 22/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 161ms/step - accuracy: 0.9017 - loss:
```

```
0.3035 - val_accuracy: 0.6650 - val_loss: 1.3399
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step - accuracy: 0.7400 - loss:
0.9209
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 90ms/step
Total training time: 682.28 seconds
Test accuracy: 0.7475
Test loss: 0.8698
```






```
<keras.src.callbacks.history.History at 0x7dc690fee410>
```

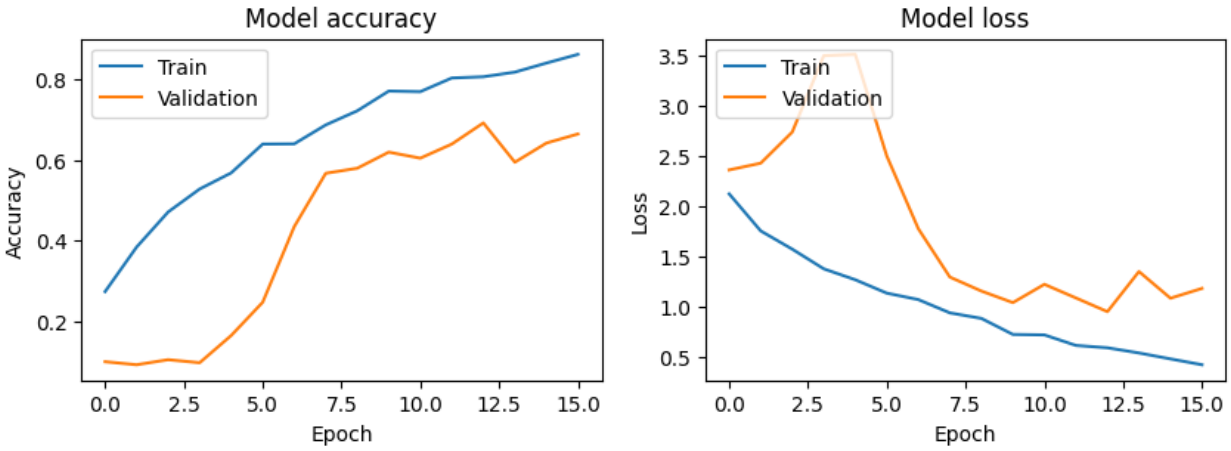
Total training time: 682.28 seconds Test accuracy: 0.7475 Test loss: 0.8698

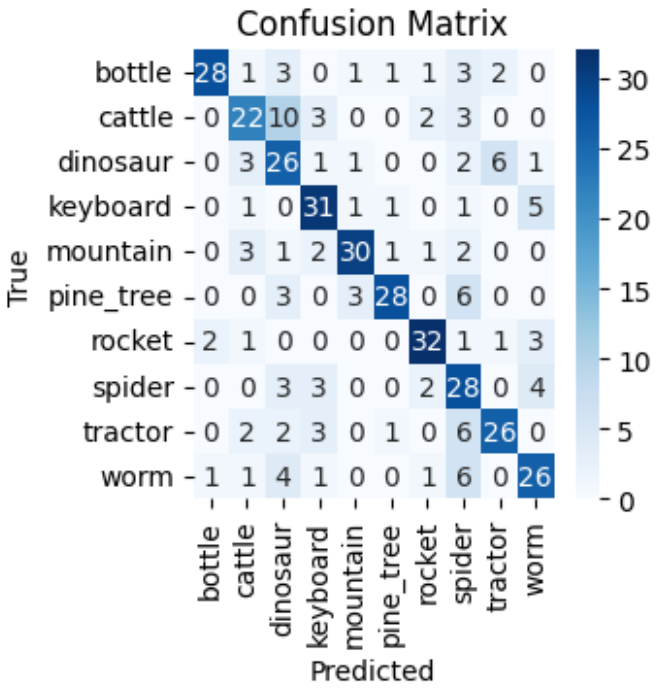
Best one so far

```
cnn6(activation='swish', dense_units=64, epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 34s 405ms/step - accuracy: 0.2264 - loss:
2.3113 - val_accuracy: 0.1000 - val_loss: 2.3639
Epoch 2/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 317ms/step - accuracy: 0.3785 - loss:
1.7762 - val_accuracy: 0.0925 - val_loss: 2.4309
Epoch 3/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 340ms/step - accuracy: 0.4685 - loss:
1.6185 - val_accuracy: 0.1050 - val_loss: 2.7403
Epoch 4/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 325ms/step - accuracy: 0.5191 - loss:
1.3893 - val_accuracy: 0.0975 - val_loss: 3.5004
Epoch 5/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 343ms/step - accuracy: 0.5688 - loss:
1.3008 - val_accuracy: 0.1650 - val_loss: 3.5130
Epoch 6/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 319ms/step - accuracy: 0.6373 - loss:
1.1620 - val_accuracy: 0.2475 - val_loss: 2.5021
Epoch 7/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 348ms/step - accuracy: 0.6399 - loss:
1.0629 - val_accuracy: 0.4350 - val_loss: 1.7792
Epoch 8/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 39s 321ms/step - accuracy: 0.6905 - loss:
0.9507 - val_accuracy: 0.5675 - val_loss: 1.2981
Epoch 9/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 345ms/step - accuracy: 0.7123 - loss:
0.8965 - val_accuracy: 0.5800 - val_loss: 1.1592
Epoch 10/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 337ms/step - accuracy: 0.7746 - loss:
0.7433 - val_accuracy: 0.6200 - val_loss: 1.0439
Epoch 11/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 40s 325ms/step - accuracy: 0.7846 - loss:
0.7046 - val_accuracy: 0.6050 - val_loss: 1.2261
Epoch 12/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 42s 347ms/step - accuracy: 0.8037 - loss:
0.6294 - val_accuracy: 0.6400 - val_loss: 1.0903
Epoch 13/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 342ms/step - accuracy: 0.8223 - loss:
0.5660 - val_accuracy: 0.6925 - val_loss: 0.9540
Epoch 14/25
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 338ms/step - accuracy: 0.8220 - loss:
0.5392 - val_accuracy: 0.5950 - val_loss: 1.3532
Epoch 15/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 21s 342ms/step - accuracy: 0.8441 - loss:
0.4970 - val_accuracy: 0.6425 - val_loss: 1.0878
Epoch 16/25
63/63 ━━━━━━━━━━━━━━━━━━━━ 41s 340ms/step - accuracy: 0.8768 - loss:
0.3815 - val_accuracy: 0.6650 - val_loss: 1.1854
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step - accuracy: 0.6788 - loss:
1.0207
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 94ms/step
Total training time: 508.03 seconds
Test accuracy: 0.6925
Test loss: 0.9540
```



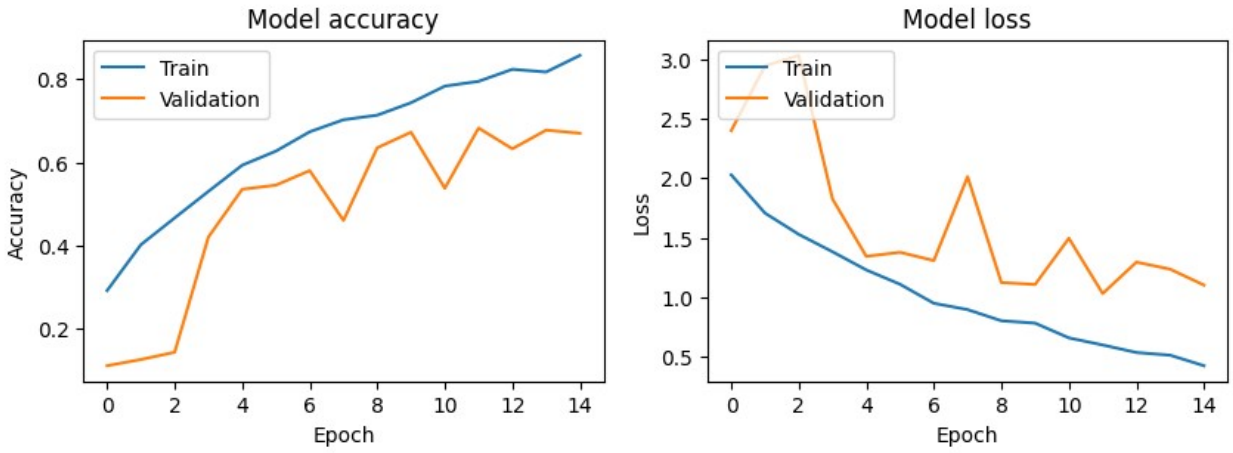


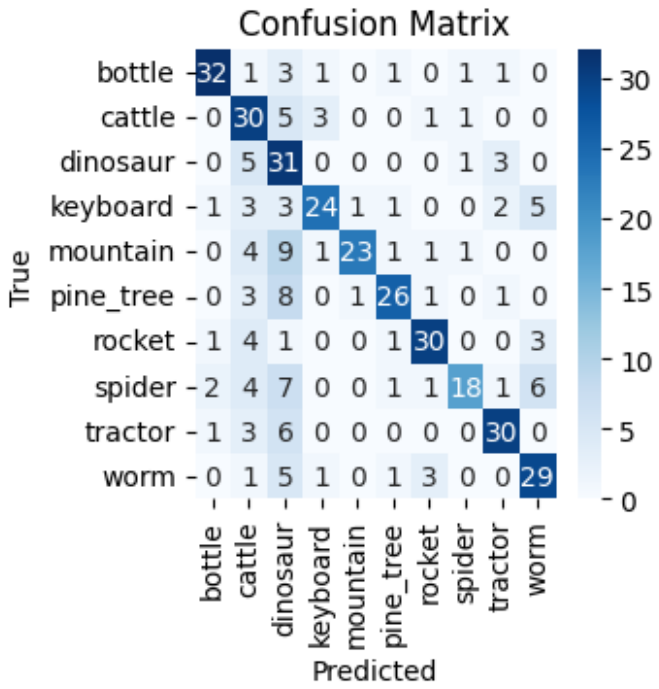

```
<keras.src.callbacks.history.History at 0x780eb9b4ea50>
```

```
cnn6(activation='swish', batch_size=16, epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 31s 180ms/step - accuracy: 0.2515 - loss:
2.2011 - val_accuracy: 0.1100 - val_loss: 2.3983
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 175ms/step - accuracy: 0.3928 - loss:
1.7417 - val_accuracy: 0.1250 - val_loss: 2.9413
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 170ms/step - accuracy: 0.4638 - loss:
1.5619 - val_accuracy: 0.1425 - val_loss: 3.0299
Epoch 4/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 176ms/step - accuracy: 0.5161 - loss:
1.4211 - val_accuracy: 0.4200 - val_loss: 1.8257
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 177ms/step - accuracy: 0.5915 - loss:
1.2267 - val_accuracy: 0.5350 - val_loss: 1.3445
Epoch 6/25
```

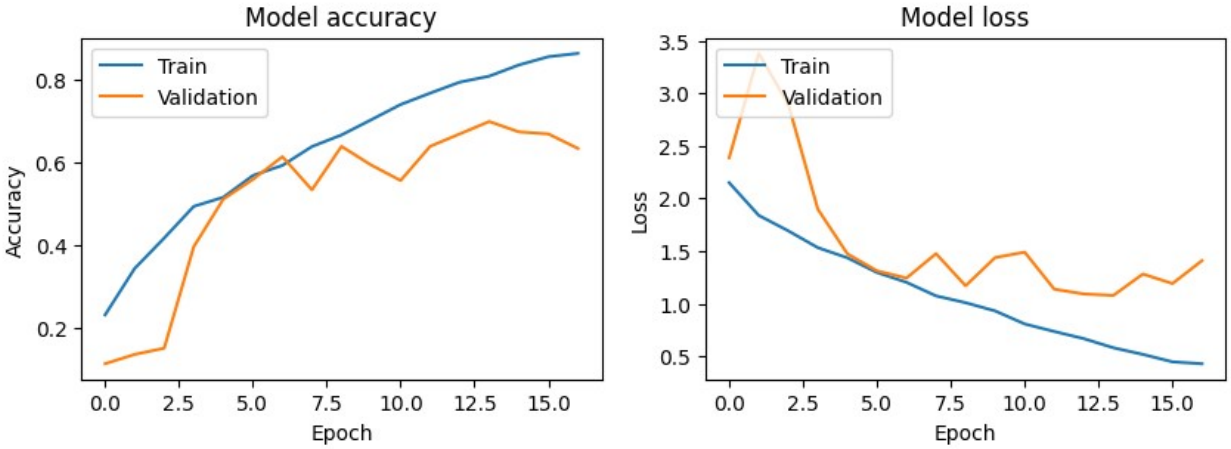
```
125/125 ━━━━━━━━━━━━━━━━━━━━ 39s 163ms/step - accuracy: 0.6195 - loss:
1.1166 - val_accuracy: 0.5450 - val_loss: 1.3785
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 176ms/step - accuracy: 0.6740 - loss:
0.9491 - val_accuracy: 0.5800 - val_loss: 1.3092
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 172ms/step - accuracy: 0.7258 - loss:
0.8448 - val_accuracy: 0.4600 - val_loss: 2.0128
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 168ms/step - accuracy: 0.7266 - loss:
0.7673 - val_accuracy: 0.6350 - val_loss: 1.1250
Epoch 10/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 175ms/step - accuracy: 0.7270 - loss:
0.8137 - val_accuracy: 0.6725 - val_loss: 1.1089
Epoch 11/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 176ms/step - accuracy: 0.8000 - loss:
0.6191 - val_accuracy: 0.5375 - val_loss: 1.4962
Epoch 12/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 165ms/step - accuracy: 0.7937 - loss:
0.6198 - val_accuracy: 0.6825 - val_loss: 1.0325
Epoch 13/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 175ms/step - accuracy: 0.8280 - loss:
0.5304 - val_accuracy: 0.6325 - val_loss: 1.2962
Epoch 14/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 176ms/step - accuracy: 0.8277 - loss:
0.4829 - val_accuracy: 0.6775 - val_loss: 1.2368
Epoch 15/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 165ms/step - accuracy: 0.8487 - loss:
0.4266 - val_accuracy: 0.6700 - val_loss: 1.1035
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.7156 - loss:
0.9481
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 96ms/step
Total training time: 582.93 seconds
Test accuracy: 0.6825
Test loss: 1.0325
```

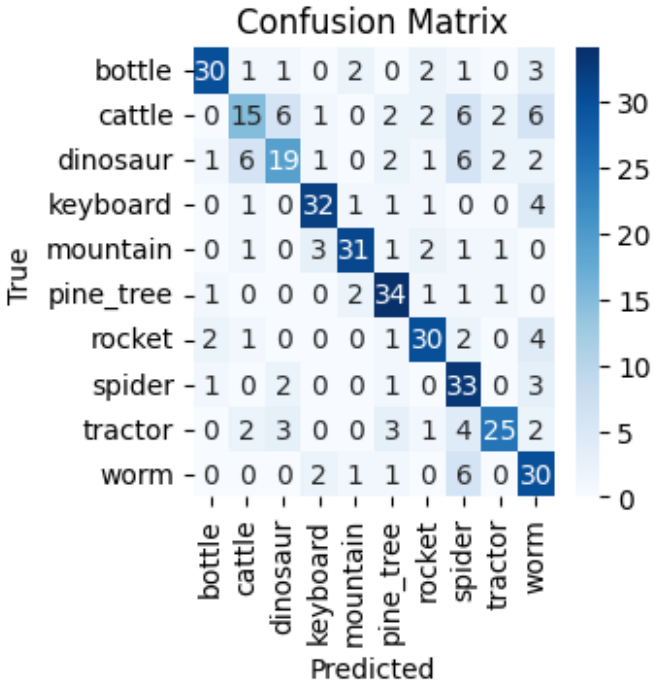





```
<keras.src.callbacks.history.History at 0x780eb2a19d10>

cnn6(activation='swish', dense_units=64, batch_size=16, conv_layers=4,
epochs=25)

/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 237ms/step - accuracy: 0.1893 - loss:
2.3119 - val_accuracy: 0.1125 - val_loss: 2.3866
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 38s 215ms/step - accuracy: 0.3214 - loss:
1.8977 - val_accuracy: 0.1350 - val_loss: 3.3780
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 216ms/step - accuracy: 0.3985 - loss:
1.7191 - val_accuracy: 0.1500 - val_loss: 2.8955
Epoch 4/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 222ms/step - accuracy: 0.4839 - loss:
1.5852 - val_accuracy: 0.3950 - val_loss: 1.8978
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 216ms/step - accuracy: 0.5052 - loss:
1.4492 - val_accuracy: 0.5100 - val_loss: 1.4739
```

```
Epoch 6/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 216ms/step - accuracy: 0.5524 - loss:
1.3163 - val_accuracy: 0.5575 - val_loss: 1.3125
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 217ms/step - accuracy: 0.5898 - loss:
1.1793 - val_accuracy: 0.6125 - val_loss: 1.2436
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 216ms/step - accuracy: 0.6700 - loss:
0.9874 - val_accuracy: 0.5325 - val_loss: 1.4757
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - accuracy: 0.6548 - loss:
1.0428 - val_accuracy: 0.6375 - val_loss: 1.1703
Epoch 10/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 215ms/step - accuracy: 0.7198 - loss:
0.8798 - val_accuracy: 0.5925 - val_loss: 1.4384
Epoch 11/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 214ms/step - accuracy: 0.7495 - loss:
0.7887 - val_accuracy: 0.5550 - val_loss: 1.4897
Epoch 12/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 216ms/step - accuracy: 0.7801 - loss:
0.6676 - val_accuracy: 0.6375 - val_loss: 1.1380
Epoch 13/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 214ms/step - accuracy: 0.7959 - loss:
0.6536 - val_accuracy: 0.6675 - val_loss: 1.0926
Epoch 14/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - accuracy: 0.8286 - loss:
0.5253 - val_accuracy: 0.6975 - val_loss: 1.0786
Epoch 15/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 215ms/step - accuracy: 0.8428 - loss:
0.5086 - val_accuracy: 0.6725 - val_loss: 1.2819
Epoch 16/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 215ms/step - accuracy: 0.8582 - loss:
0.4358 - val_accuracy: 0.6675 - val_loss: 1.1915
Epoch 17/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 27s 216ms/step - accuracy: 0.8484 - loss:
0.4616 - val_accuracy: 0.6325 - val_loss: 1.4090
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 84ms/step - accuracy: 0.6554 - loss:
1.2281
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 192ms/step
Total training time: 626.39 seconds
Test accuracy: 0.6975
Test loss: 1.0786
```



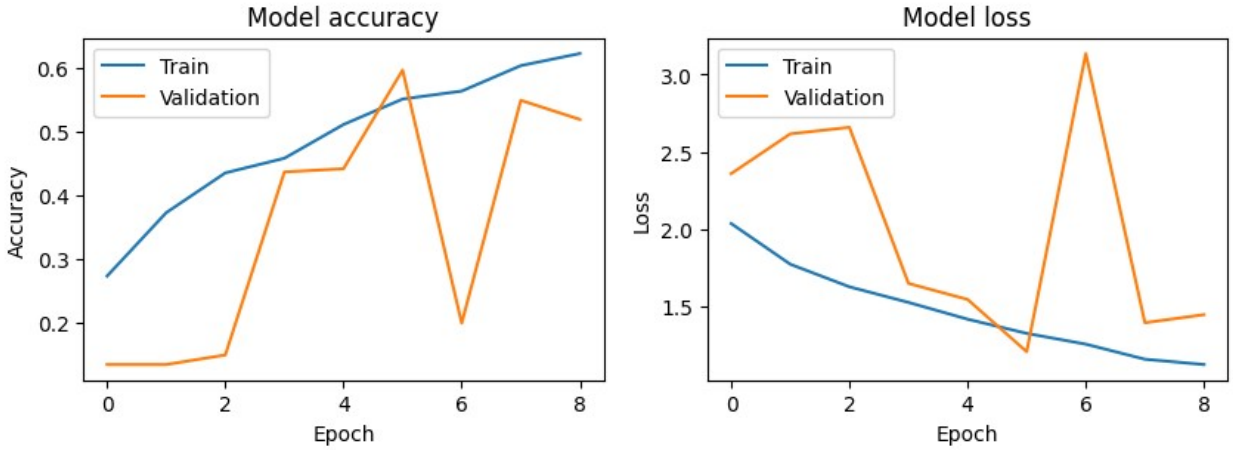


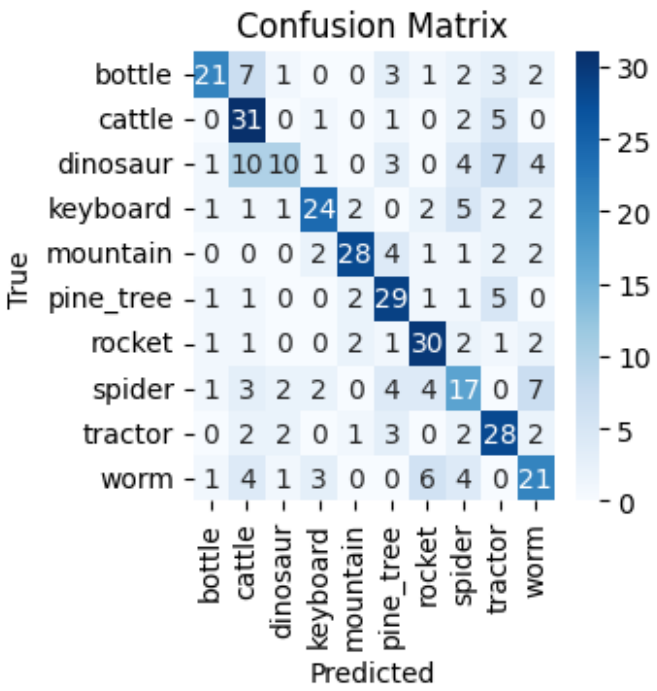

```
<keras.src.callbacks.history.History at 0x7dc6ba61e410>
```

```
cnn6(activation='swish', dense_units=64, batch_size=16, conv_layers=2,
epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 24s 129ms/step - accuracy: 0.2184 - loss:
2.2057 - val_accuracy: 0.1350 - val_loss: 2.3591
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 15s 117ms/step - accuracy: 0.3647 - loss:
1.7927 - val_accuracy: 0.1350 - val_loss: 2.6162
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 15s 122ms/step - accuracy: 0.4303 - loss:
1.6562 - val_accuracy: 0.1500 - val_loss: 2.6584
Epoch 4/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 117ms/step - accuracy: 0.4470 - loss:
1.5385 - val_accuracy: 0.4375 - val_loss: 1.6487
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 15s 121ms/step - accuracy: 0.5144 - loss:
1.3914 - val_accuracy: 0.4425 - val_loss: 1.5451
Epoch 6/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 126ms/step - accuracy: 0.5588 - loss:
1.3236 - val_accuracy: 0.5975 - val_loss: 1.2070
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 19s 116ms/step - accuracy: 0.5766 - loss:
1.2219 - val_accuracy: 0.2000 - val_loss: 3.1358
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 14s 115ms/step - accuracy: 0.6023 - loss:
1.1768 - val_accuracy: 0.5500 - val_loss: 1.3945
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 116ms/step - accuracy: 0.6119 - loss:
1.1229 - val_accuracy: 0.5200 - val_loss: 1.4465
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - accuracy: 0.5825 - loss:
1.2691
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step
Total training time: 163.78 seconds
Test accuracy: 0.5975
Test loss: 1.2070
```






```
<keras.src.callbacks.history.History at 0x7dc6abbb6410>
```

I'll use the cnn6 as my final model with dense_units=64 and batch_size=16. I'll test it again with the remainder of the CIFAR-100 image set to see how it performs.

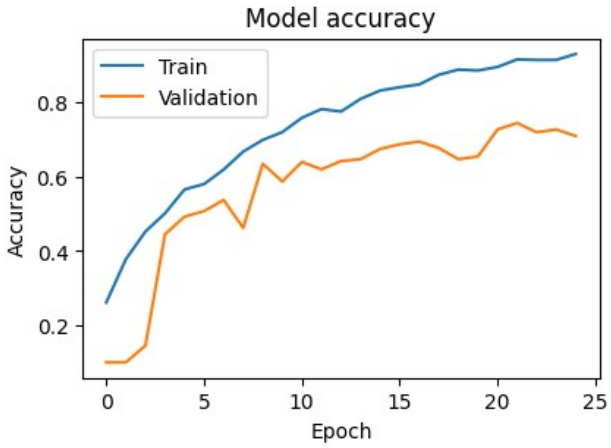
Final model

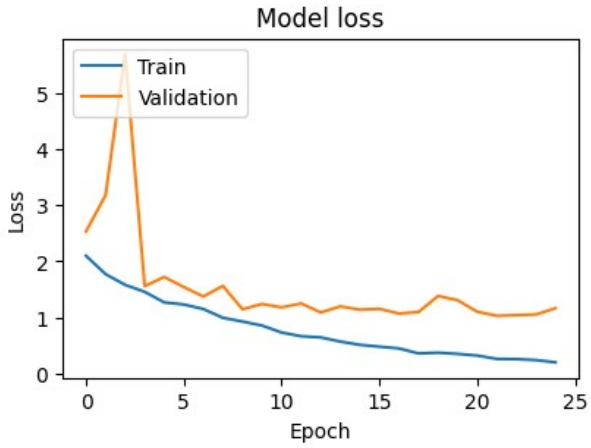
```
final_model = cnn6(activation='swish', dense_units=64, batch_size=16,
epochs=25)
```

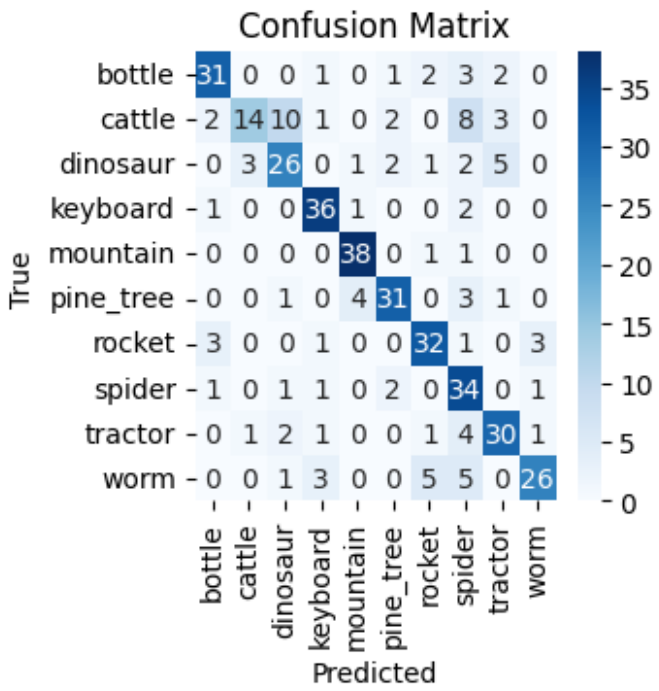
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 29s 177ms/step - accuracy: 0.2096 - loss:
2.2722 - val_accuracy: 0.1000 - val_loss: 2.5295
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 38s 155ms/step - accuracy: 0.3537 - loss:
1.8171 - val_accuracy: 0.1000 - val_loss: 3.1750
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 164ms/step - accuracy: 0.4541 - loss:
1.5834 - val_accuracy: 0.1450 - val_loss: 5.6900
Epoch 4/25
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 159ms/step - accuracy: 0.4975 - loss:
1.4571 - val_accuracy: 0.4450 - val_loss: 1.5569
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 165ms/step - accuracy: 0.5711 - loss:
1.2396 - val_accuracy: 0.4925 - val_loss: 1.7202
Epoch 6/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 168ms/step - accuracy: 0.5969 - loss:
1.1613 - val_accuracy: 0.5075 - val_loss: 1.5409
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 168ms/step - accuracy: 0.6241 - loss:
1.1291 - val_accuracy: 0.5375 - val_loss: 1.3723
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 164ms/step - accuracy: 0.6638 - loss:
0.9914 - val_accuracy: 0.4625 - val_loss: 1.5609
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 157ms/step - accuracy: 0.6936 - loss:
0.9285 - val_accuracy: 0.6350 - val_loss: 1.1459
Epoch 10/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 168ms/step - accuracy: 0.6966 - loss:
0.8714 - val_accuracy: 0.5875 - val_loss: 1.2391
Epoch 11/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 159ms/step - accuracy: 0.7657 - loss:
0.7189 - val_accuracy: 0.6400 - val_loss: 1.1795
Epoch 12/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 167ms/step - accuracy: 0.8110 - loss:
0.6153 - val_accuracy: 0.6200 - val_loss: 1.2491
Epoch 13/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.7704 - loss:
0.6548 - val_accuracy: 0.6425 - val_loss: 1.0886
Epoch 14/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 158ms/step - accuracy: 0.8242 - loss:
0.5316 - val_accuracy: 0.6475 - val_loss: 1.1968
Epoch 15/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 164ms/step - accuracy: 0.8571 - loss:
0.4450 - val_accuracy: 0.6750 - val_loss: 1.1387
Epoch 16/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 158ms/step - accuracy: 0.8285 - loss:
0.4865 - val_accuracy: 0.6875 - val_loss: 1.1525
Epoch 17/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 161ms/step - accuracy: 0.8492 - loss:
0.4525 - val_accuracy: 0.6950 - val_loss: 1.0700
Epoch 18/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 164ms/step - accuracy: 0.8905 - loss:
0.3314 - val_accuracy: 0.6775 - val_loss: 1.0965
Epoch 19/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 168ms/step - accuracy: 0.8967 - loss:
0.3338 - val_accuracy: 0.6475 - val_loss: 1.3824
Epoch 20/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 39s 154ms/step - accuracy: 0.8813 - loss:
```

```
0.3461 - val_accuracy: 0.6550 - val_loss: 1.3064
Epoch 21/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 167ms/step - accuracy: 0.9065 - loss:
0.2863 - val_accuracy: 0.7275 - val_loss: 1.1011
Epoch 22/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 158ms/step - accuracy: 0.9075 - loss:
0.2624 - val_accuracy: 0.7450 - val_loss: 1.0297
Epoch 23/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 167ms/step - accuracy: 0.9311 - loss:
0.2094 - val_accuracy: 0.7200 - val_loss: 1.0415
Epoch 24/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 157ms/step - accuracy: 0.9125 - loss:
0.2519 - val_accuracy: 0.7275 - val_loss: 1.0524
Epoch 25/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 164ms/step - accuracy: 0.9260 - loss:
0.2108 - val_accuracy: 0.7100 - val_loss: 1.1641
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step - accuracy: 0.7112 - loss:
1.2517
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 91ms/step
Total training time: 764.88 seconds
Test accuracy: 0.7450
Test loss: 1.0297
```







Next will be saving the model and printing out the summary to get a better view of what the model is made out of and also to see the number of parameters in the model.

```
final_model.model.save('final_model.keras')

model = tf.keras.models.load_model('final_model.keras')
model.summary()
```

```
Model: "sequential_17"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_104 (Conv2D)                  │ (None, 32, 32, 32)          │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_121              │ (None, 32, 32, 32)          │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_105 (Conv2D)                  │ (None, 32, 32, 32)          │           9,248 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
```

```
──────────────────┤
│ batch_normalization_122               │ (None, 32, 32, 32)          │
128 │
│ (BatchNormalization)                  │                             │
    │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ max_pooling2d_52 (MaxPooling2D)       │ (None, 16, 16, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ dropout_69 (Dropout)                  │ (None, 16, 16, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ conv2d_106 (Conv2D)                   │ (None, 16, 16, 64)          │
18,496 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ batch_normalization_123               │ (None, 16, 16, 64)          │
256 │
│ (BatchNormalization)                  │                             │
    │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ conv2d_107 (Conv2D)                   │ (None, 16, 16, 64)          │
36,928 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ batch_normalization_124               │ (None, 16, 16, 64)          │
256 │
│ (BatchNormalization)                  │                             │
    │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ max_pooling2d_53 (MaxPooling2D)       │ (None, 8, 8, 64)            │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ dropout_70 (Dropout)                  │ (None, 8, 8, 64)            │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ conv2d_108 (Conv2D)                   │ (None, 8, 8, 128)           │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
│ batch_normalization_125               │ (None, 8, 8, 128)           │
512 │
```

```
| (BatchNormalization)                  |                              |
|                                       |                              |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| conv2d_109 (Conv2D)                   | (None, 8, 8, 128)            |
147,584 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| batch_normalization_126               | (None, 8, 8, 128)            |
512 |
| (BatchNormalization)                  |                              |
|                                       |                              |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| max_pooling2d_54 (MaxPooling2D)       | (None, 4, 4, 128)            |
0 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| dropout_71 (Dropout)                  | (None, 4, 4, 128)            |
0 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| global_average_pooling2d_17           | (None, 128)                  |
0 |
| (GlobalAveragePooling2D)              |                              |
|                                       |                              |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| dense_51 (Dense)                      | (None, 64)                   |
8,256 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| batch_normalization_127               | (None, 64)                   |
256 |
| (BatchNormalization)                  |                              |
|                                       |                              |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| dense_52 (Dense)                      | (None, 32)                   |
2,080 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| dropout_72 (Dropout)                  | (None, 32)                   |
0 |
├───────────────────────────────────────┼──────────────────────────────┼
──────────────────┤
| dense_53 (Dense)                      | (None, 10)                   |
330 |
```

```
└─────────────────────────────────────────────┴──────────────────────────────┴──
─────────────────┘

 Total params: 598,422 (2.28 MB)

 Trainable params: 298,698 (1.14 MB)

 Non-trainable params: 1,024 (4.00 KB)

 Optimizer params: 298,700 (1.14 MB)
```

Here is the summary of the final model. It consists of: 6 Convolutional Layers 3 Max Pooling Layers 7 times Batch Normalization 4 times Dropout 1 Global Average Pooling 3 Dense Layers

Total params: 598,422 Trainable params: 298,698 Non-trainable params: 1,024 Optimizer params: 298,700

Testing with new images from same classes

```
# Here I'm taking all the unused images from the training set for the
selected classes and testing the model with them

x_final_test, y_final_test = [], []

for class_id in classes:
    # Indices of all images in the training set
    all_train_indices = np.where(y_train.flatten() == class_id)[0]

    # Indices of images NOT initially used for training
    final_test_indices = all_train_indices[num_images:]

    # Add the not used images to the new test set
    x_final_test.extend(x_train[final_test_indices])
    y_final_test.extend(y_train[final_test_indices])

# Convert lists to NumPy arrays
x_final_test = np.array(x_final_test)
y_final_test = np.array(y_final_test)

# Create a mapping from original class labels to new labels (0 to 9)
class_mapping = {class_id: i for i, class_id in enumerate(classes)}

# Map original class labels to new labels
y_final_test = np.vectorize(class_mapping.get)(y_final_test)

# One-hot encode the labels
y_final_test = tf.keras.utils.to_categorical(y_final_test,
len(classes))

# Normalize the image data
x_final_test = x_final_test.astype("float32") / 255
```

```
test_loss, test_acc = model.evaluate(x_final_test, y_final_test)
print(f"Test accuracy on unseen data: {test_acc:.4f}")
print(f"Test loss on unseen data: {test_loss:.4f}")
```

```
94/94 ━━━━━━━━━━━━━━━━━━━━ 9s 76ms/step - accuracy: 0.7064 - loss:
1.0902
Test accuracy on unseen data: 0.7260
Test loss on unseen data: 0.9612
```

Final accuracy of the model when testing with images that weren't used in the training or validation is 0.7260. I'll now try the model with new image classes as well to see how it generalizes with different images, because now it's optimized with the initially chosen classes.

Testing with new image classes

```
# This is basically just a copy of the code from the beginning of the
project.
# This could've been done using a function but I don't mind having to
use the same code again this way.

# Using 10 classes and 200 images per class
classes_new = [0,10,20,30,40,50,60,70,80,90]

# Names of the selected classes
selected_class_names_new = [class_names[i] for i in classes_new]

# Create the test and training sets
x_train_new, y_train_new = [], []
x_test_new, y_test_new = [], []

for class_id in classes_new:
    train_indices_new = np.where(y_train.flatten() == class_id)[0]
[:num_images]
    test_indices_new = np.where(y_test.flatten() == class_id)[0]
[:num_images // 5]

    x_train_new.append(x_train[train_indices_new])
    y_train_new.append(y_train[train_indices_new])
    x_test_new.append(x_test[test_indices_new])
    y_test_new.append(y_test[test_indices_new])

# Convert lists to NumPy arrays
x_train_new = np.concatenate(x_train_new, axis=0)
y_train_new = np.concatenate(y_train_new, axis=0)
x_test_new = np.concatenate(x_test_new, axis=0)
y_test_new = np.concatenate(y_test_new, axis=0)

# Create a mapping from original class labels to new labels (0 to 9)
class_mapping_new = {class_id: i for i, class_id in
enumerate(classes_new)}
```

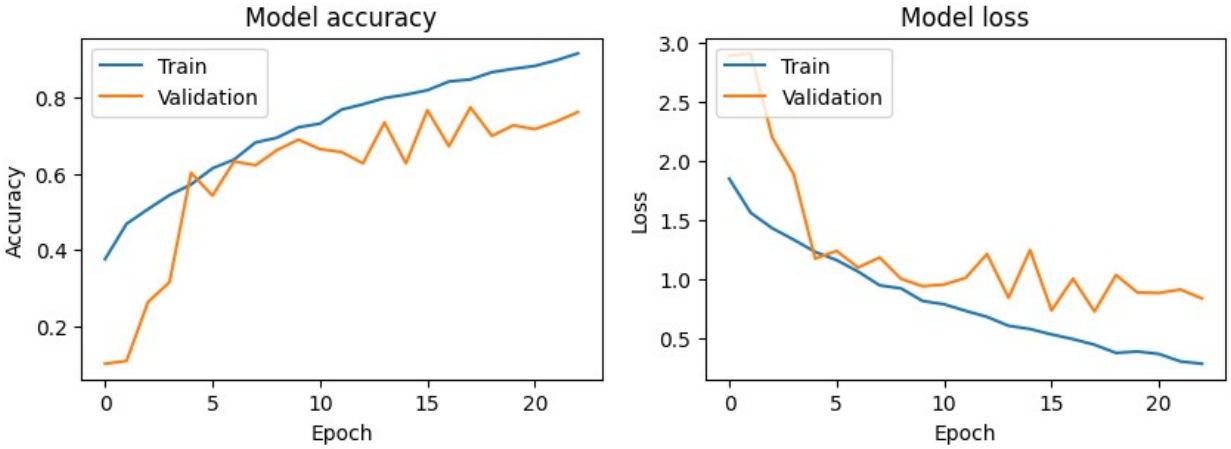
```
# Map original class labels to new labels
y_train_new = np.vectorize(class_mapping_new.get)(y_train_new)
y_test_new = np.vectorize(class_mapping_new.get)(y_test_new)

# One-hot encode the labels
y_train_new = tf.keras.utils.to_categorical(y_train_new,
len(classes_new))
y_test_new = tf.keras.utils.to_categorical(y_test_new,
len(classes_new))

# Normalize the image data
x_train_new = x_train_new.astype("float32") / 255
x_test_new = x_test_new.astype("float32") / 255

# Display one image from each class
fig, axes = plt.subplots(1, len(classes_new), figsize=(15, 3))

for i, class_id in enumerate(classes_new):
    index = np.where(y_train.flatten() == class_id)[0][0]  # Select
first occurrence
    image = x_train[index]

    axes[i].imshow(image)
    axes[i].set_title(class_names[class_id])
    axes[i].axis("off")

plt.show()
```

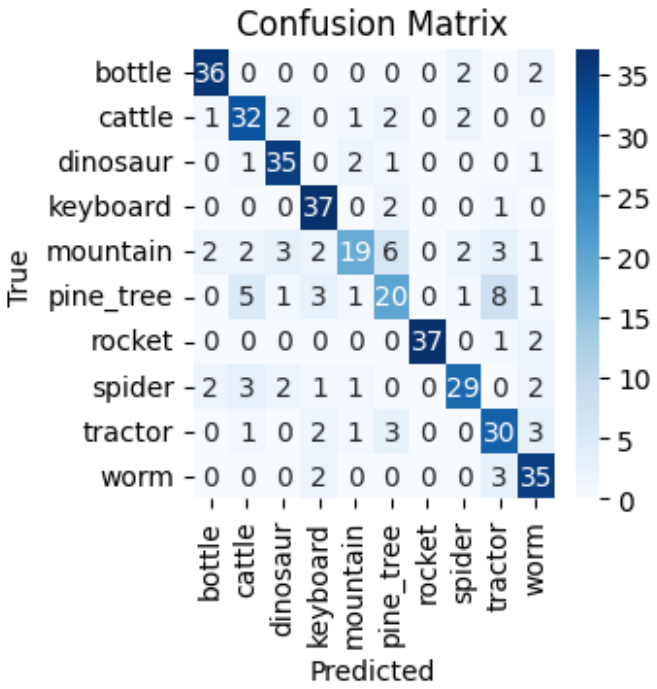

```
cnn6(train_data=x_train_new,
        train_labels=y_train_new,
        val_data=x_test_new,
        val_labels=y_test_new,
        activation='swish',
        dense_units=64,
        batch_size=16,
        epochs=25)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Epoch 1/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 31s 179ms/step - accuracy: 0.3064 - loss:
2.0232 - val_accuracy: 0.1000 - val_loss: 2.8918
Epoch 2/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 169ms/step - accuracy: 0.4611 - loss:
1.5864 - val_accuracy: 0.1075 - val_loss: 2.9134
Epoch 3/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 39s 157ms/step - accuracy: 0.4942 - loss:
1.4753 - val_accuracy: 0.2625 - val_loss: 2.2013
Epoch 4/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 169ms/step - accuracy: 0.5417 - loss:
1.3259 - val_accuracy: 0.3150 - val_loss: 1.8892
Epoch 5/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 160ms/step - accuracy: 0.5648 - loss:
1.2649 - val_accuracy: 0.6025 - val_loss: 1.1716
Epoch 6/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 169ms/step - accuracy: 0.5964 - loss:
1.1851 - val_accuracy: 0.5425 - val_loss: 1.2367
Epoch 7/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 160ms/step - accuracy: 0.6422 - loss:
1.0520 - val_accuracy: 0.6325 - val_loss: 1.0958
Epoch 8/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 167ms/step - accuracy: 0.6719 - loss:
0.9430 - val_accuracy: 0.6225 - val_loss: 1.1808
Epoch 9/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 160ms/step - accuracy: 0.6989 - loss:
0.9122 - val_accuracy: 0.6625 - val_loss: 0.9995
Epoch 10/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 170ms/step - accuracy: 0.7054 - loss:
0.8621 - val_accuracy: 0.6900 - val_loss: 0.9372
Epoch 11/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 20s 160ms/step - accuracy: 0.7381 - loss:
0.7689 - val_accuracy: 0.6650 - val_loss: 0.9516
Epoch 12/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 169ms/step - accuracy: 0.7834 - loss:
0.6934 - val_accuracy: 0.6575 - val_loss: 1.0074
Epoch 13/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 39s 156ms/step - accuracy: 0.7826 - loss:
0.6473 - val_accuracy: 0.6275 - val_loss: 1.2104
Epoch 14/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 24s 181ms/step - accuracy: 0.8060 - loss:
0.6061 - val_accuracy: 0.7350 - val_loss: 0.8389
Epoch 15/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 23s 183ms/step - accuracy: 0.8065 - loss:
0.5574 - val_accuracy: 0.6275 - val_loss: 1.2447
Epoch 16/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 39s 169ms/step - accuracy: 0.8153 - loss:
0.5442 - val_accuracy: 0.7675 - val_loss: 0.7310
Epoch 17/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 161ms/step - accuracy: 0.8596 - loss:
```

```
0.4289 - val_accuracy: 0.6725 - val_loss: 1.0009
Epoch 18/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 169ms/step - accuracy: 0.8462 - loss:
0.4343 - val_accuracy: 0.7750 - val_loss: 0.7228
Epoch 19/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 41s 168ms/step - accuracy: 0.8587 - loss:
0.3974 - val_accuracy: 0.7000 - val_loss: 1.0327
Epoch 20/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 160ms/step - accuracy: 0.8930 - loss:
0.3310 - val_accuracy: 0.7275 - val_loss: 0.8837
Epoch 21/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 22s 169ms/step - accuracy: 0.8736 - loss:
0.3759 - val_accuracy: 0.7175 - val_loss: 0.8798
Epoch 22/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 21s 172ms/step - accuracy: 0.8962 - loss:
0.3024 - val_accuracy: 0.7375 - val_loss: 0.9091
Epoch 23/25
125/125 ━━━━━━━━━━━━━━━━━━━━ 40s 167ms/step - accuracy: 0.9171 - loss:
0.3038 - val_accuracy: 0.7625 - val_loss: 0.8337
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step - accuracy: 0.8082 - loss:
0.6628
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 92ms/step
Total training time: 688.73 seconds
Test accuracy: 0.7750
Test loss: 0.7228
```





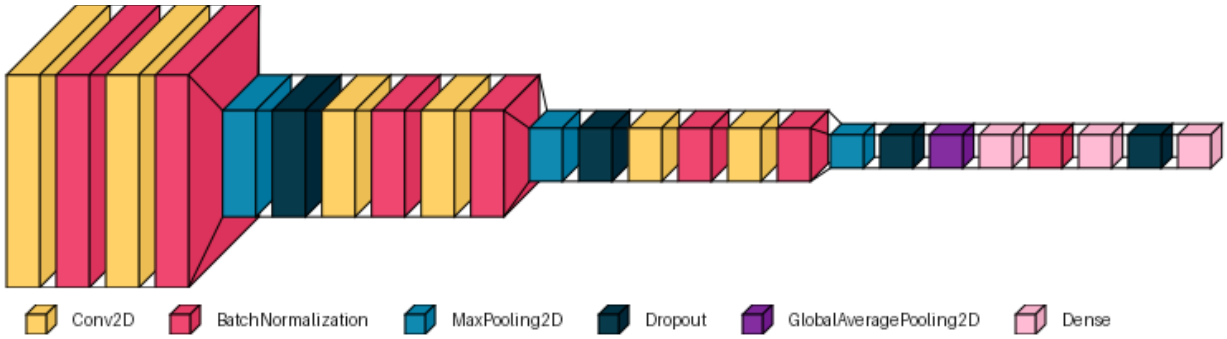

```
<keras.src.callbacks.history.History at 0x7dc686596410>
```

The model works very well with other image classes aswell obtaining even higher validation accuracy than before with the original classes. Also there is little to none overfitting especially compared with many of the previous iterations of the model with many different hyperparameter combinations.

Visualizing the model

```
# Visualization of the model as a picture that shows different layers

import visualkeras
from PIL import Image

# Create and save the visualization
visualkeras.layered_view(model, legend=True,
to_file='model_visualization.png').save('model_visualization.png')

# Display the image
display(Image.open('model_visualization.png'))
```

```
/usr/local/lib/python3.11/dist-packages/visualkeras/layered.py:86:
UserWarning: The legend_text_spacing_offset parameter is deprecated
and will be removed in a future release.
  warnings.warn("The legend_text_spacing_offset parameter is
deprecated and will be removed in a future release.")
```


Conv2D
BatchNormalization
MaxPooling2D
Dropout
GlobalAveragePooling2D
Dense