# AI-570 - Assignment # 2

July 1, 2024

[ ]:
```
#!/usr/bin/env python
```

## 1 Zoo classifier

- Build a classifier to classify images onto one of three classes/labels. The dataset Download datasetconsists of three folders (cats, dogs and panda), containing images in different dimensions. Chose one of the following options to build your classifier.

**- Option 1:** Build a model based on a convolutional neural network. For a classification problem, you usually choose softmax as output activation function and the categorical_crossentropy as a loss function.

Notes:

- As in all deep neural networks, it is difficult to find out the optimal CNN model. That's why we usually build one or more neural network models with different architectures (i.e., different number of layers, different activation functions and/or different number of neurons per layer) and different hyperparameter values (i.e., learning rates, number of epochs, batch size, optimizer (Adam, RMSProp, SGD) ). For this assignment, start by building a base line model and fit it to your training and testing datasets. Based on the evaluation of your loss/accuracy plots, you may decide to improve the model accuracy by varying the neural network architecture and/or hyper-parameter values ( please refer to the lessons and their case-studies in Keras for examples).
- Regardless your choice of option 1 or option 2, try to improve your the accuracy of model. Another important point is the data exploration. Before training your model, you should get insights from your dataset (number of images, view show couples of images, and specify the image size to use in the training) . This is an indispensable and systematic task in any data analytics or machine learning projects.

[4]:
```
#
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from glob import glob
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout
from tensorflow.keras.optimizers import Adam
```

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
Collecting tensorflow
  Downloading tensorflow-2.16.2-cp312-cp312-macosx_12_0_arm64.whl.metadata (4.1
kB)
Collecting absl-py>=1.0.0 (from tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow)
  Using cached flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow)
  Downloading gast-0.6.0.tar.gz (27 kB)
  Preparing metadata (setup.py) … done
Collecting google-pasta>=0.1.1 (from tensorflow)
  Using cached google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Requirement already satisfied: h5py>=3.10.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (3.11.0)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-1-py2.py3-none-macosx_11_0_arm64.whl.metadata (5.2
kB)
Collecting ml-dtypes~=0.3.1 (from tensorflow)
  Downloading ml_dtypes-0.3.2-cp312-cp312-macosx_10_9_universal2.whl.metadata
(20 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow)
  Using cached opt_einsum-3.3.0-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: packaging in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (23.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (2.32.2)
Requirement already satisfied: setuptools in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (69.5.1)
Requirement already satisfied: six>=1.12.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow)
  Using cached termcolor-2.4.0-py3-none-any.whl.metadata (6.1 kB)
```

```
Requirement already satisfied: typing-extensions>=3.6.6 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (1.14.1)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow)
  Downloading grpcio-1.64.1-cp312-cp312-macosx_10_9_universal2.whl.metadata (3.3
kB)
Collecting tensorboard<2.17,>=2.16 (from tensorflow)
  Using cached tensorboard-2.16.2-py3-none-any.whl.metadata (1.6 kB)
Collecting keras>=3.0.0 (from tensorflow)
  Downloading keras-3.4.1-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: numpy<2.0.0,>=1.26.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from astunparse>=1.6.0->tensorflow) (0.43.0)
Requirement already satisfied: rich in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from keras>=3.0.0->tensorflow) (13.3.5)
Collecting namex (from keras>=3.0.0->tensorflow)
  Using cached namex-0.0.8-py3-none-any.whl.metadata (246 bytes)
Collecting optree (from keras>=3.0.0->tensorflow)
  Downloading optree-0.11.0-cp312-cp312-macosx_11_0_arm64.whl.metadata (45 kB)
                           45.4/45.4 kB
4.5 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from requests<3,>=2.21.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from requests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from requests<3,>=2.21.0->tensorflow) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from requests<3,>=2.21.0->tensorflow) (2024.6.2)
Requirement already satisfied: markdown>=2.6.8 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from tensorboard<2.17,>=2.16->tensorflow) (3.4.1)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from
tensorboard<2.17,>=2.16->tensorflow)
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
```

```
packages (from tensorboard<2.17,>=2.16->tensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from rich->keras>=3.0.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from rich->keras>=3.0.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages (from markdown-it-py<3.0.0,>=2.2.0->rich->keras>=3.0.0->tensorflow)
(0.1.0)
Downloading tensorflow-2.16.2-cp312-cp312-macosx_12_0_arm64.whl (227.1 MB)
                         227.1/227.1 MB
18.7 MB/s eta 0:00:0000:0100:01
Using cached absl_py-2.1.0-py3-none-any.whl (133 kB)
Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Using cached flatbuffers-24.3.25-py2.py3-none-any.whl (26 kB)
Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Downloading grpcio-1.64.1-cp312-cp312-macosx_10_9_universal2.whl (10.3 MB)
                         10.3/10.3 MB
33.9 MB/s eta 0:00:00 0:00:01
Downloading keras-3.4.1-py3-none-any.whl (1.1 MB)
                         1.1/1.1 MB
27.8 MB/s eta 0:00:0000:01
Downloading libclang-18.1.1-1-py2.py3-none-macosx_11_0_arm64.whl (25.8 MB)
                         25.8/25.8 MB
25.4 MB/s eta 0:00:0000:0100:01
Downloading ml_dtypes-0.3.2-cp312-cp312-macosx_10_9_universal2.whl (393
kB)
                         393.6/393.6 kB
21.1 MB/s eta 0:00:00
Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Using cached tensorboard-2.16.2-py3-none-any.whl (5.5 MB)
Using cached termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Using cached namex-0.0.8-py3-none-any.whl (5.8 kB)
Downloading optree-0.11.0-cp312-cp312-macosx_11_0_arm64.whl (276 kB)
                         276.7/276.7 kB
22.1 MB/s eta 0:00:00
Building wheels for collected packages: gast
  Building wheel for gast (setup.py) … done
  Created wheel for gast: filename=gast-0.6.0-py3-none-any.whl size=21172
sha256=7cbcdfd700cae2700de321cf78b9f269fdd02ec5d981d97e2824d1ceeb15a4a3
  Stored in directory: /Users/zelalemabahana/Library/Caches/pip/wheels/42/3f/6a/
b8ddfb8e9453cabbe4c01b13aa03a200bfaa3aa24013d38924
```

```
Successfully built gast
Installing collected packages: namex, libclang, flatbuffers, termcolor,
tensorboard-data-server, optree, opt-einsum, ml-dtypes, grpcio, google-pasta,
gast, astunparse, absl-py, tensorboard, keras, tensorflow
Successfully installed absl-py-2.1.0 astunparse-1.6.3 flatbuffers-24.3.25
gast-0.6.0 google-pasta-0.2.0 grpcio-1.64.1 keras-3.4.1 libclang-18.1.1 ml-
dtypes-0.3.2 namex-0.0.8 opt-einsum-3.3.0 optree-0.11.0 tensorboard-2.16.2
tensorboard-data-server-0.7.2 tensorflow-2.16.2 termcolor-2.4.0
```

## 2 Step 1: Explore the data

- Before diving into model building, let start by loading the dataset, inspecting the number of images in each class, and visualizing some sample images.

[1]:
```python
# path to the dataset folders
base_path = '/Users/zelalemabahana/Desktop/PennState/PSU-AI570/images'
categories = ['cats', 'dogs', 'panda']

# Count the number of images in each category
for category in categories:
    folder_path = os.path.join(base_path, category)
    num_images = len(glob(os.path.join(folder_path, '*.jpg')))
    print(f'Number of images in {category}: {num_images}')

# Visualize some images from each category
def visualize_images(base_path, categories):
    fig, axes = plt.subplots(1, len(categories), figsize=(15, 5))
    for ax, category in zip(axes, categories):
        img_path = glob(os.path.join(base_path, category, '*.jpg'))[0]
        img = mpimg.imread(img_path)
        ax.imshow(img)
        ax.set_title(category)
        ax.axis('off')
    plt.show()

visualize_images(base_path, categories)
```

```
Number of images in cats: 1000
Number of images in dogs: 1000
Number of images in panda: 1000
```


# 3 Step 2: Data Preprocessing

- Let us Resize the images to a uniform size (150x150x3) and create training and validation datasets using Keras' ImageDataGenerator.

[21]:
```
# Define image size and batch size
img_size = (150, 150)
batch_size = 32

# Create an ImageDataGenerator for data augmentation and normalization
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 2400 images belonging to 3 classes.
```

```
Found 600 images belonging to 3 classes.
```

# 4 Step 3: Let us build CNN Models and Test Accuracy for various models

### 4.0.1 Model #1

[14]:
```python
def build_and_train_model(train_generator, validation_generator,
                          conv_layers=[32, 64, 128],
                          dense_units=512,
                          dropout_rate=0.5,
                          learning_rate=0.001,
                          optimizer_type='adam',
                          epochs=10):
    """
    The following set up will build, compile, and train a CNN model with␣
 ↪specified hyperparameters.

    Parameters:
    - train_generator: The training data generator
    - validation_generator: The validation data generator
    - conv_layers: List of integers, number of filters for each Conv2D layer
    - dense_units: Integer, number of units in the Dense layer
    - dropout_rate: Float, dropout rate for the Dropout layer
    - learning_rate: Float, learning rate for the optimizer
    - optimizer_type: String, type of optimizer ('adam', 'rmsprop', 'sgd')
    - epochs: Integer, number of training epochs

    Returns:
    - model: The trained Keras model
    - history: Training history object
    """
    # Build the model
    model = Sequential()
    model.add(Conv2D(conv_layers[0], (3, 3), activation='relu',␣
 ↪input_shape=(150, 150, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    for filters in conv_layers[1:]:
        model.add(Conv2D(filters, (3, 3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(3, activation='softmax'))
```

```python
    # Select optimizer
    if optimizer_type == 'adam':
        optimizer = Adam(learning_rate=learning_rate)
    elif optimizer_type == 'rmsprop':
        optimizer = RMSprop(learning_rate=learning_rate)
    elif optimizer_type == 'sgd':
        optimizer = SGD(learning_rate=learning_rate)
    else:
        raise ValueError("Invalid optimizer type")

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

    # Train and evaluat the model
    history = model.fit(
        train_generator,
        epochs=epochs,
        validation_data=validation_generator
    )

    loss, accuracy = model.evaluate(validation_generator)
    print(f'Validation Loss: {loss}')
    print(f'Validation Accuracy: {accuracy}')

    # Plot training & validation accuracy and loss
    plot_history(history)

    return model, history

def plot_history(history):
    plt.style.use('ggplot')  # Use the ggplot style
    plt.figure(figsize=(14, 6))

    # training & validation accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy', marker='o')
    plt.plot(history.history['val_accuracy'], label='Val Accuracy', marker='o')
    plt.xlabel('Epochs', fontsize=12)
    plt.ylabel('Accuracy', fontsize=12)
    plt.legend(loc='upper left')
    plt.title('Accuracy over Epochs', fontsize=14)
    plt.grid(True)

    # training & validation loss
    plt.subplot(1, 2, 2)
```

```
    plt.plot(history.history['loss'], label='Train Loss', marker='o')
    plt.plot(history.history['val_loss'], label='Val Loss', marker='o')
    plt.xlabel('Epochs', fontsize=12)
    plt.ylabel('Loss', fontsize=12)
    plt.legend(loc='upper right')
    plt.title('Loss over Epochs', fontsize=14)
    plt.grid(True)

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

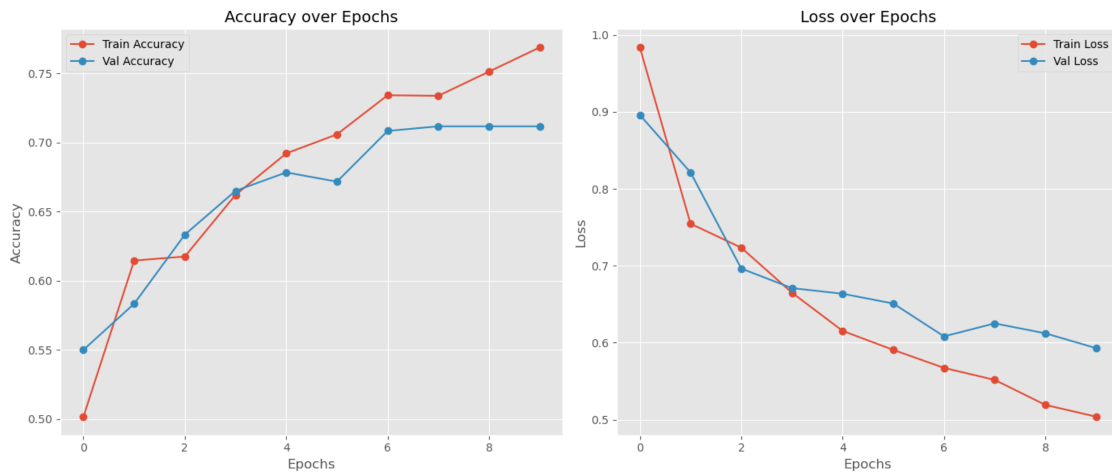
# 5 Model #1

[16]:
```
# calling the function with different hyperparameters
model, history = build_and_train_model(
    train_generator=train_generator,
    validation_generator=validation_generator,
    conv_layers=[32, 64, 128],
    dense_units=512,
    dropout_rate=0.5,
    learning_rate=0.001,
    optimizer_type='adam',
    epochs=10
)
```

```
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
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 1/10
75/75              22s 275ms/step -
accuracy: 0.4482 - loss: 1.1827 - val_accuracy: 0.5500 - val_loss: 0.8960
Epoch 2/10
75/75              21s 273ms/step -
accuracy: 0.6178 - loss: 0.7756 - val_accuracy: 0.5833 - val_loss: 0.8213
Epoch 3/10
75/75              21s 269ms/step -
accuracy: 0.6211 - loss: 0.7258 - val_accuracy: 0.6333 - val_loss: 0.6964
Epoch 4/10
75/75              21s 272ms/step -
accuracy: 0.6517 - loss: 0.6807 - val_accuracy: 0.6650 - val_loss: 0.6710
Epoch 5/10
75/75              21s 267ms/step -
accuracy: 0.6998 - loss: 0.6098 - val_accuracy: 0.6783 - val_loss: 0.6637
```

```
Epoch 6/10
75/75              21s 266ms/step -
accuracy: 0.7046 - loss: 0.6014 - val_accuracy: 0.6717 - val_loss: 0.6509
Epoch 7/10
75/75              21s 266ms/step -
accuracy: 0.7255 - loss: 0.5721 - val_accuracy: 0.7083 - val_loss: 0.6083
Epoch 8/10
75/75              21s 273ms/step -
accuracy: 0.7187 - loss: 0.5623 - val_accuracy: 0.7117 - val_loss: 0.6251
Epoch 9/10
75/75              22s 273ms/step -
accuracy: 0.7571 - loss: 0.5151 - val_accuracy: 0.7117 - val_loss: 0.6122
Epoch 10/10
75/75              21s 270ms/step -
accuracy: 0.7873 - loss: 0.4756 - val_accuracy: 0.7117 - val_loss: 0.5930
19/19              3s 143ms/step -
accuracy: 0.7508 - loss: 0.5394
Validation Loss: 0.5603217482566833
Validation Accuracy: 0.7366666793823242
```




# 6 Model #2

- increase the number of epochs to 20 – More epochs to train the deeper model
- Increase the dropout rate to .6 – to prevent overfitting
- decrease the learning rate to 0.0001 – can help for finer adjustments to the weights
- change the convolutional layer – to increased number of filters

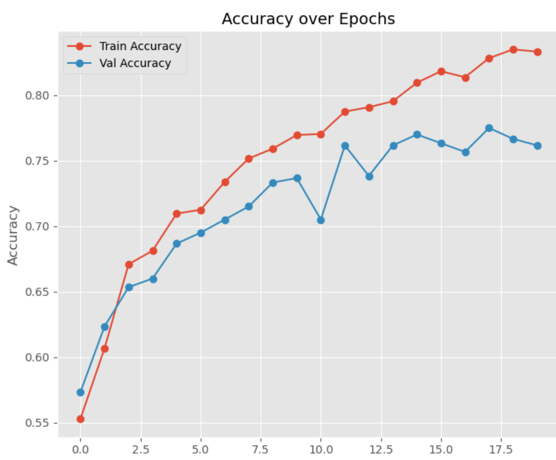
[18]:
```
# calling the function with different hyperparameters
model, history = build_and_train_model(
    train_generator=train_generator,
    validation_generator=validation_generator,
```

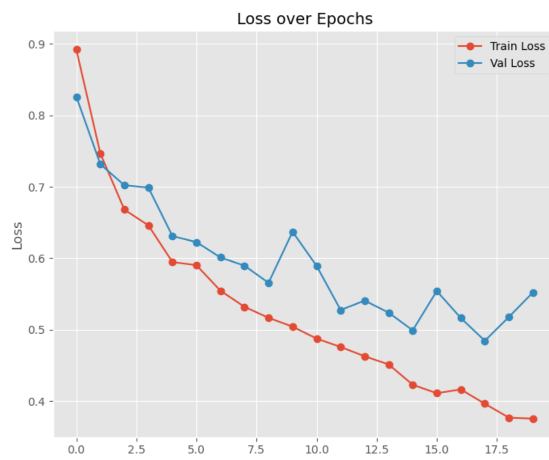
```
    conv_layers=[64, 128, 256],
    dense_units=512,
    dropout_rate=0.6,
    learning_rate=0.0001,
    optimizer_type='adam',
    epochs=20
)
```

```
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
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
75/75                 55s 716ms/step -
accuracy: 0.5115 - loss: 0.9745 - val_accuracy: 0.5733 - val_loss: 0.8263
Epoch 2/20
75/75                 54s 704ms/step -
accuracy: 0.6013 - loss: 0.7640 - val_accuracy: 0.6233 - val_loss: 0.7316
Epoch 3/20
75/75                 87s 1s/step -
accuracy: 0.6448 - loss: 0.7118 - val_accuracy: 0.6533 - val_loss: 0.7021
Epoch 4/20
75/75                 64s 833ms/step -
accuracy: 0.6789 - loss: 0.6404 - val_accuracy: 0.6600 - val_loss: 0.6989
Epoch 5/20
75/75                 134s 2s/step -
accuracy: 0.7046 - loss: 0.5939 - val_accuracy: 0.6867 - val_loss: 0.6310
Epoch 6/20
75/75                 130s 2s/step -
accuracy: 0.7206 - loss: 0.5886 - val_accuracy: 0.6950 - val_loss: 0.6225
Epoch 7/20
75/75                 127s 2s/step -
accuracy: 0.7272 - loss: 0.5619 - val_accuracy: 0.7050 - val_loss: 0.6010
Epoch 8/20
75/75                 133s 2s/step -
accuracy: 0.7484 - loss: 0.5313 - val_accuracy: 0.7150 - val_loss: 0.5895
Epoch 9/20
75/75                 134s 2s/step -
accuracy: 0.7606 - loss: 0.5283 - val_accuracy: 0.7333 - val_loss: 0.5655
Epoch 10/20
75/75                 129s 2s/step -
accuracy: 0.7804 - loss: 0.4992 - val_accuracy: 0.7367 - val_loss: 0.6370
Epoch 11/20
75/75                 203s 3s/step -
accuracy: 0.7748 - loss: 0.4891 - val_accuracy: 0.7050 - val_loss: 0.5892
```

```
Epoch 12/20
75/75             134s 2s/step -
accuracy: 0.7814 - loss: 0.5004 - val_accuracy: 0.7617 - val_loss: 0.5274
Epoch 13/20
75/75             125s 2s/step -
accuracy: 0.8088 - loss: 0.4282 - val_accuracy: 0.7383 - val_loss: 0.5406
Epoch 14/20
75/75             52s 676ms/step -
accuracy: 0.7839 - loss: 0.4739 - val_accuracy: 0.7617 - val_loss: 0.5237
Epoch 15/20
75/75             52s 675ms/step -
accuracy: 0.8106 - loss: 0.4254 - val_accuracy: 0.7700 - val_loss: 0.4990
Epoch 16/20
75/75             52s 676ms/step -
accuracy: 0.8177 - loss: 0.4051 - val_accuracy: 0.7633 - val_loss: 0.5540
Epoch 17/20
75/75             50s 661ms/step -
accuracy: 0.8104 - loss: 0.4152 - val_accuracy: 0.7567 - val_loss: 0.5163
Epoch 18/20
75/75             51s 673ms/step -
accuracy: 0.8257 - loss: 0.4021 - val_accuracy: 0.7750 - val_loss: 0.4843
Epoch 19/20
75/75             52s 676ms/step -
accuracy: 0.8377 - loss: 0.3757 - val_accuracy: 0.7667 - val_loss: 0.5176
Epoch 20/20
75/75             53s 694ms/step -
accuracy: 0.8229 - loss: 0.3768 - val_accuracy: 0.7617 - val_loss: 0.5520
19/19             4s 196ms/step -
accuracy: 0.7462 - loss: 0.6094
Validation Loss: 0.5710324645042419
Validation Accuracy: 0.7666666507720947
```



- Model #2 showed a slight improvement over model #1 where the accuracy was lifted from .73 to .76

# 7 Model #3

- For model #3, let us try a more aggressive data augmentation to make the model more robust.
- We will add parameters to the image generator by adding rotation, width and height shift
- withh the updated data augmentation, let us check if our model specification in Model 1 shows improvement in performance

[19]:
```
# Define image size and batch size
img_size = (150, 150)
batch_size = 32

# Create an ImageDataGenerator for data augmentation and normalization
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    rotation_range=30,  # Added rotation
    width_shift_range=0.2,  # Added width shift
    height_shift_range=0.2,  # Added height shift
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 2400 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
```

[20]:
```python
# calling the function with different hyperparameters
model, history = build_and_train_model(
    train_generator=train_generator,
    validation_generator=validation_generator,
    conv_layers=[32, 64, 128],
    dense_units=512,
    dropout_rate=0.5,
    learning_rate=0.001,
    optimizer_type='adam',
    epochs=10
)
```

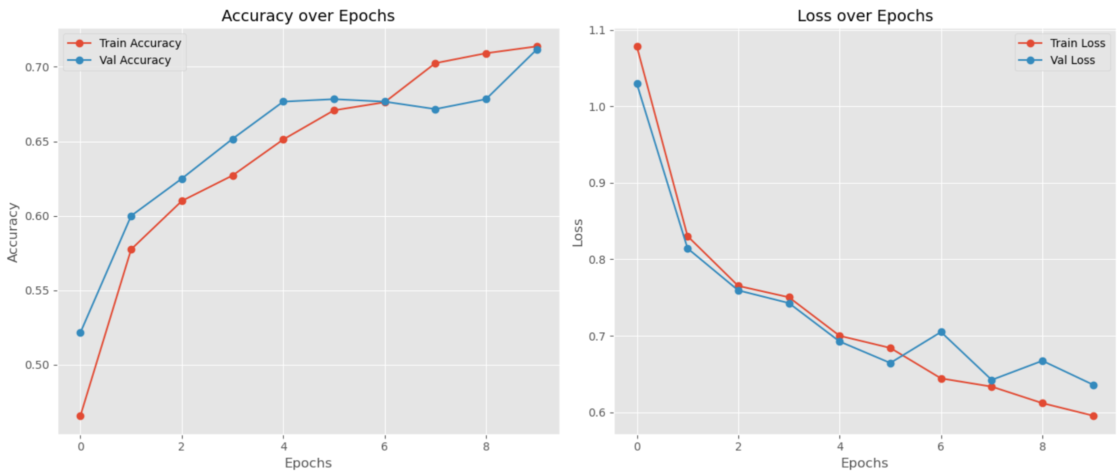
```
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Epoch 1/10

/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

75/75          23s 285ms/step -
accuracy: 0.4038 - loss: 1.4120 - val_accuracy: 0.5217 - val_loss: 1.0294
Epoch 2/10
75/75          23s 289ms/step -
accuracy: 0.5650 - loss: 0.8536 - val_accuracy: 0.6000 - val_loss: 0.8143
Epoch 3/10
75/75          22s 276ms/step -
accuracy: 0.5948 - loss: 0.7847 - val_accuracy: 0.6250 - val_loss: 0.7594
Epoch 4/10
75/75          21s 266ms/step -
accuracy: 0.6136 - loss: 0.7698 - val_accuracy: 0.6517 - val_loss: 0.7428
Epoch 5/10
75/75          21s 271ms/step -
accuracy: 0.6513 - loss: 0.6772 - val_accuracy: 0.6767 - val_loss: 0.6926
Epoch 6/10
75/75          21s 272ms/step -
accuracy: 0.6764 - loss: 0.6733 - val_accuracy: 0.6783 - val_loss: 0.6645
Epoch 7/10
75/75          21s 266ms/step -
accuracy: 0.6807 - loss: 0.6398 - val_accuracy: 0.6767 - val_loss: 0.7050
```

```
Epoch 8/10
75/75              21s 264ms/step -
accuracy: 0.6954 - loss: 0.6426 - val_accuracy: 0.6717 - val_loss: 0.6423
Epoch 9/10
75/75              21s 265ms/step -
accuracy: 0.7084 - loss: 0.6239 - val_accuracy: 0.6783 - val_loss: 0.6674
Epoch 10/10
75/75              21s 266ms/step -
accuracy: 0.7190 - loss: 0.5872 - val_accuracy: 0.7117 - val_loss: 0.6361
19/19              3s 144ms/step -
accuracy: 0.6990 - loss: 0.6620
Validation Loss: 0.6473731398582458
Validation Accuracy: 0.699999988079071
```




- With the data augmentation, the accuracy for model #1 got worse; therefore, we will stay away from aggressive data augmentation

# 8 Model #4

- Let us try to make improvemnts over the specification over Model #2
- Increase the Number of Filters: By increasing the number of filters in each convolutional layer, we can allow the model to learn more complex features.
- Increase the Number of Dense Units: More units in the dense layer can help the model learn more complex representations.
- Higher Dropout Rates for Complex Models: If you add more layers and units, consider increasing the dropout rate.
- Decrease the Learning Rate: Lower learning rates can help in fine-tuning the model more precisely, especially if we have a deeper network.

- Increase the Number of Epochs: Allow the model to train longer to potentially achieve better convergence.

[24]:
```
# calling the function with different hyperparameters
model, history = build_and_train_model(
    train_generator=train_generator,
    validation_generator=validation_generator,
    conv_layers = [64, 128, 256, 512],
    dense_units=1024,
    dropout_rate=0.6,
    learning_rate=0.00005,
    optimizer_type='adam',
    epochs=30
)
```

```
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
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)
```

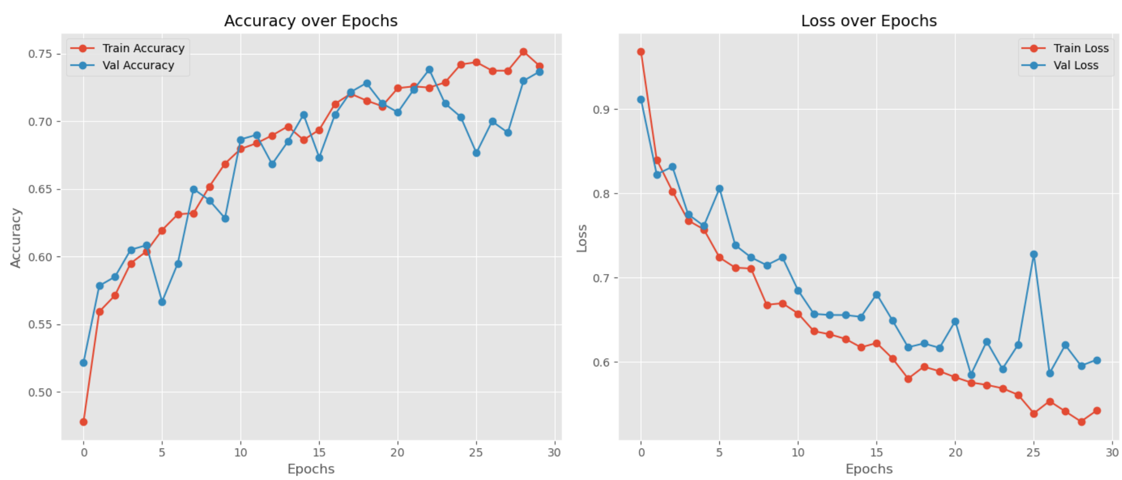
```
Epoch 1/30
```

```
/Users/zelalemabahana/Desktop/PennState/DAAN862/anaconda3/lib/python3.12/site-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

```
75/75                64s 836ms/step -
accuracy: 0.4215 - loss: 1.0388 - val_accuracy: 0.5217 - val_loss: 0.9115
Epoch 2/30
75/75                65s 854ms/step -
accuracy: 0.5425 - loss: 0.8462 - val_accuracy: 0.5783 - val_loss: 0.8224
Epoch 3/30
75/75                63s 825ms/step -
accuracy: 0.5791 - loss: 0.7964 - val_accuracy: 0.5850 - val_loss: 0.8315
Epoch 4/30
75/75                63s 825ms/step -
accuracy: 0.5830 - loss: 0.7761 - val_accuracy: 0.6050 - val_loss: 0.7751
Epoch 5/30
75/75                64s 846ms/step -
accuracy: 0.6021 - loss: 0.7452 - val_accuracy: 0.6083 - val_loss: 0.7612
Epoch 6/30
75/75                63s 830ms/step -
accuracy: 0.6004 - loss: 0.7537 - val_accuracy: 0.5667 - val_loss: 0.8058
```

```
Epoch 7/30
75/75            63s 826ms/step -
accuracy: 0.6230 - loss: 0.7193 - val_accuracy: 0.5950 - val_loss: 0.7384
Epoch 8/30
75/75            62s 820ms/step -
accuracy: 0.6401 - loss: 0.6913 - val_accuracy: 0.6500 - val_loss: 0.7240
Epoch 9/30
75/75            63s 831ms/step -
accuracy: 0.6412 - loss: 0.6792 - val_accuracy: 0.6417 - val_loss: 0.7147
Epoch 10/30
75/75            64s 836ms/step -
accuracy: 0.6612 - loss: 0.6910 - val_accuracy: 0.6283 - val_loss: 0.7239
Epoch 11/30
75/75            63s 823ms/step -
accuracy: 0.6819 - loss: 0.6565 - val_accuracy: 0.6867 - val_loss: 0.6848
Epoch 12/30
75/75            65s 852ms/step -
accuracy: 0.6724 - loss: 0.6472 - val_accuracy: 0.6900 - val_loss: 0.6569
Epoch 13/30
75/75            63s 823ms/step -
accuracy: 0.6927 - loss: 0.6312 - val_accuracy: 0.6683 - val_loss: 0.6555
Epoch 14/30
75/75            63s 823ms/step -
accuracy: 0.7164 - loss: 0.6006 - val_accuracy: 0.6850 - val_loss: 0.6555
Epoch 15/30
75/75            64s 836ms/step -
accuracy: 0.6903 - loss: 0.6085 - val_accuracy: 0.7050 - val_loss: 0.6534
Epoch 16/30
75/75            68s 888ms/step -
accuracy: 0.6990 - loss: 0.6153 - val_accuracy: 0.6733 - val_loss: 0.6803
Epoch 17/30
75/75            64s 845ms/step -
accuracy: 0.7221 - loss: 0.6121 - val_accuracy: 0.7050 - val_loss: 0.6493
Epoch 18/30
75/75            64s 839ms/step -
accuracy: 0.7186 - loss: 0.5680 - val_accuracy: 0.7217 - val_loss: 0.6173
Epoch 19/30
75/75            67s 881ms/step -
accuracy: 0.7223 - loss: 0.5901 - val_accuracy: 0.7283 - val_loss: 0.6218
Epoch 20/30
75/75            63s 826ms/step -
accuracy: 0.7183 - loss: 0.5790 - val_accuracy: 0.7133 - val_loss: 0.6166
Epoch 21/30
75/75            66s 870ms/step -
accuracy: 0.7165 - loss: 0.5868 - val_accuracy: 0.7067 - val_loss: 0.6484
Epoch 22/30
75/75            63s 828ms/step -
accuracy: 0.7352 - loss: 0.5731 - val_accuracy: 0.7233 - val_loss: 0.5850
```

```
Epoch 23/30
75/75            64s 835ms/step -
accuracy: 0.7290 - loss: 0.5645 - val_accuracy: 0.7383 - val_loss: 0.6241
Epoch 24/30
75/75            90s 1s/step -
accuracy: 0.7167 - loss: 0.5671 - val_accuracy: 0.7133 - val_loss: 0.5916
Epoch 25/30
75/75            106s 1s/step -
accuracy: 0.7570 - loss: 0.5531 - val_accuracy: 0.7033 - val_loss: 0.6203
Epoch 26/30
75/75            111s 1s/step -
accuracy: 0.7396 - loss: 0.5382 - val_accuracy: 0.6767 - val_loss: 0.7278
Epoch 27/30
75/75            160s 2s/step -
accuracy: 0.7356 - loss: 0.5576 - val_accuracy: 0.7000 - val_loss: 0.5863
Epoch 28/30
75/75            153s 2s/step -
accuracy: 0.7339 - loss: 0.5553 - val_accuracy: 0.6917 - val_loss: 0.6199
Epoch 29/30
75/75            144s 2s/step -
accuracy: 0.7477 - loss: 0.5407 - val_accuracy: 0.7300 - val_loss: 0.5954
Epoch 30/30
75/75            151s 2s/step -
accuracy: 0.7182 - loss: 0.5595 - val_accuracy: 0.7367 - val_loss: 0.6024
19/19            11s 577ms/step -
accuracy: 0.7316 - loss: 0.5855
Validation Loss: 0.5812419652938843
Validation Accuracy: 0.7416666746139526
```



# 9 Conclusions:

- Overall, Model #2 is the best model in terms of model accuracy (0.76).
- To enhance the models' accuracy, I focused on both data augmentation and model parameter tuning. I started with data exploration understand the dataset, including analyzing the number of images per class, visualizing sample images, and defining the image size for training. This step was essential to make informed decisions. I then optimized my model by adjusting data augmentation techniques and tuning hyperparameters, to achive an improved model performance.

[ ]:
```

```