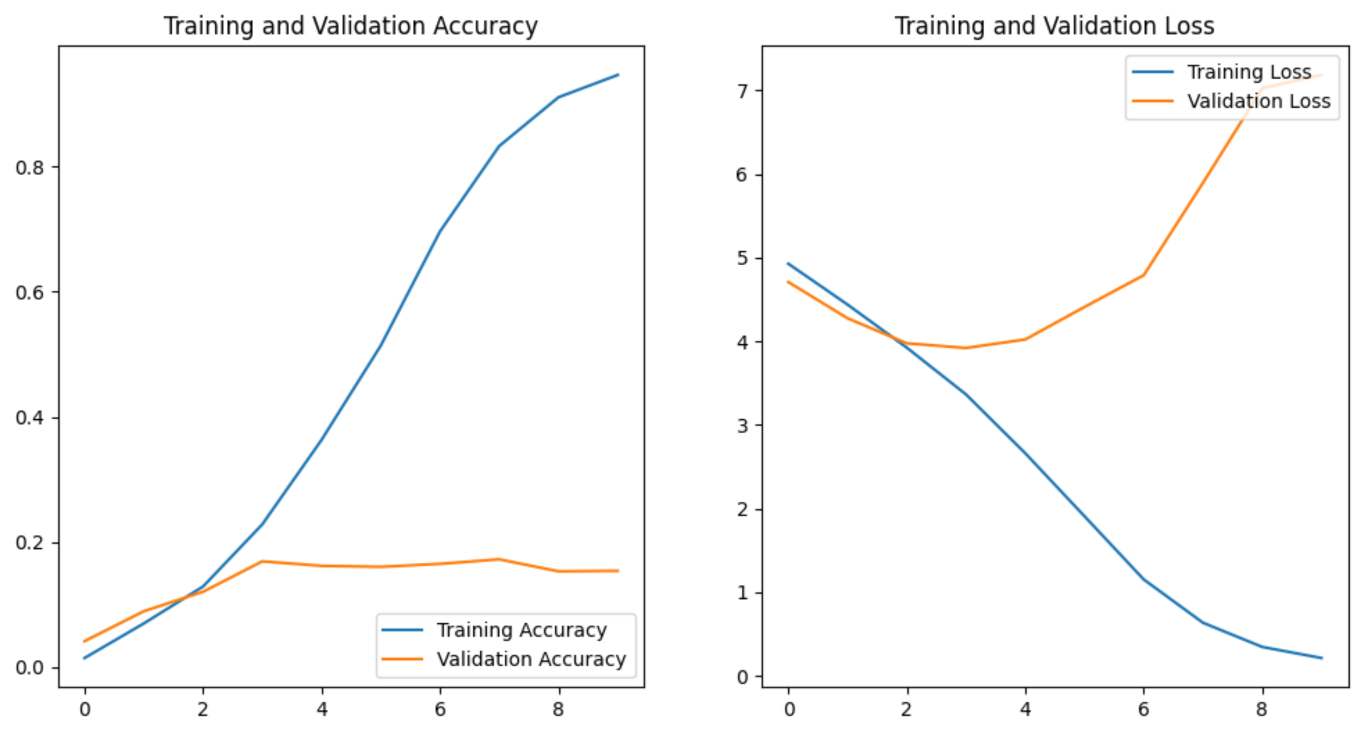
Project Overview

The goal of this project is to create a model capable of correctly identifying the animals when given an image.

First Attempt

Defined a Convolutional Neural Network. Also added layers for rescaling to normalize, convolutional layers to extract features, max pooling to reduce layers and retain the most important information, flattening to make it easier to process information, and dense layers to find relationships. This run topped out at 17.22% validation accuracy, with a clear sign of overfitting.

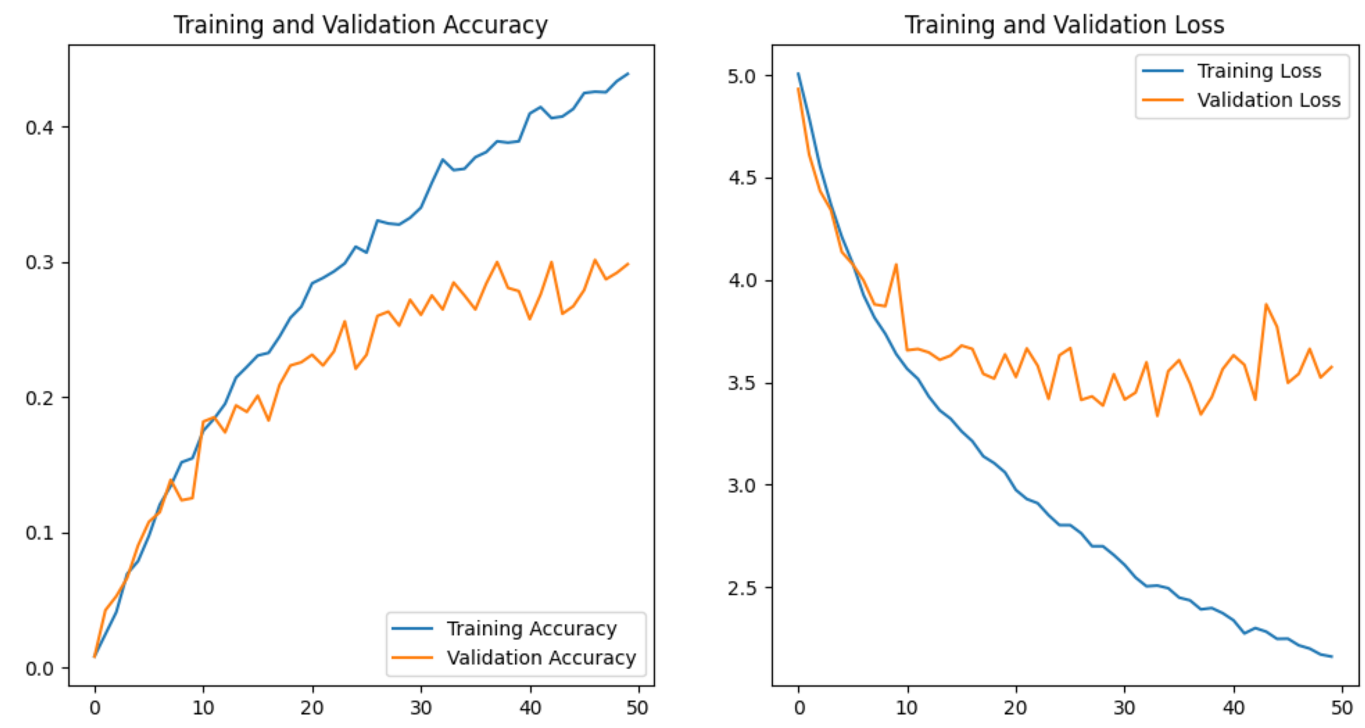
```
model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(151, activation='softmax')
])

model.summary()

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

epochs = 10
history = model.fit(
    train,
    validation_data=val,
    epochs=epochs
)
```



Improvements Through Augmentation

Added image augmentation to reduce overfitting so easily. Augmentations included random flips, rotations, brightness changes, contrast adjustments, translations, and zooms. This helps the model learn from a more varied set of examples too, which is important with this limited dataset. This run topped out at 30.14% validation accuracy.

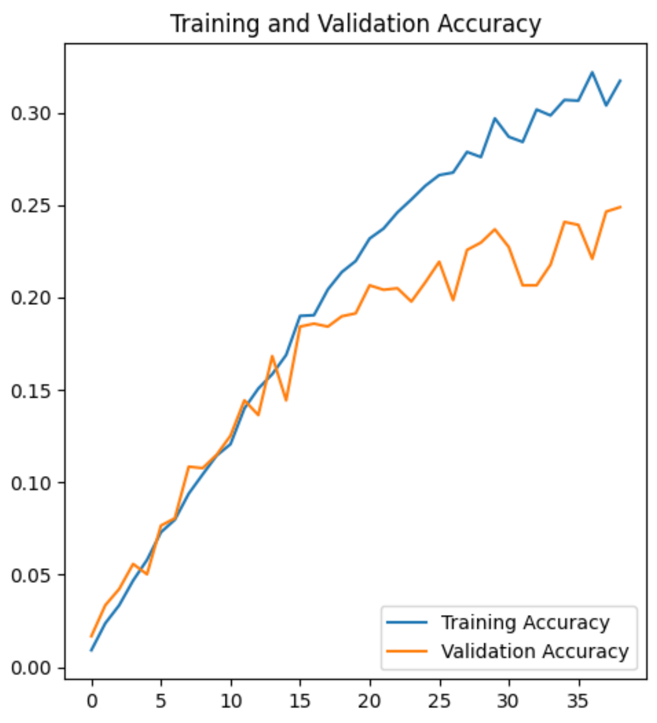
```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

aug = tf.keras.Sequential([
  layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
  layers.RandomRotation(0.1),
  layers.RandomBrightness(0.2),
  layers.RandomContrast(0.2),
  layers.RandomTranslation(height_factor=0.1, width_factor=0.1),
  layers.RandomZoom(0.1)
])

model = Sequential([
    aug,
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(train_ds.class_names), activation='softmax')
])

model.summary()

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

epochs = 50

history = model.fit(
    train,
    validation_data=val,
    epochs=epochs
)
```

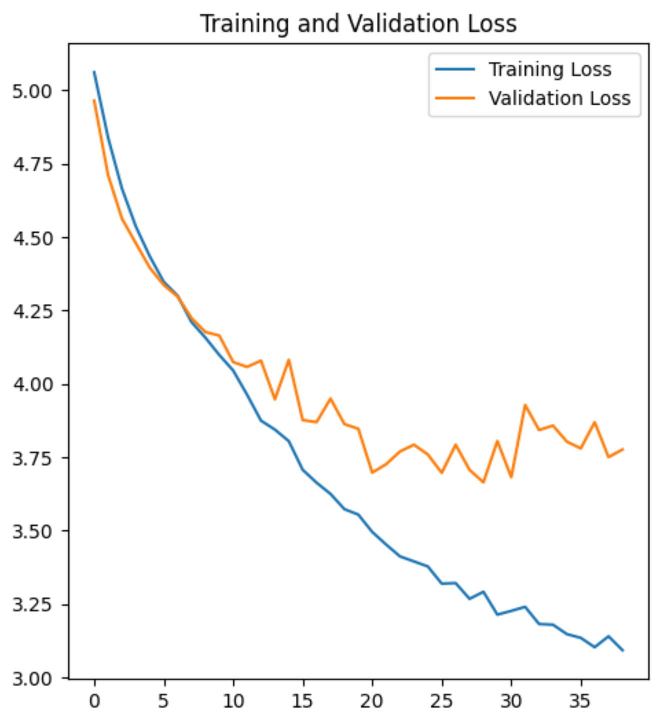


More Aggressive Data Augmentation

This section created an even more varied augmentation for images, to try and reduce overfitting even more. Also attempted Batch Normalization to also help reduce overfitting, as well as ideally speeding up the training process leading to a faster convergence. Also added dropout layers to help generalize the data better. However, with a validation accuracy being 26.40%, it actually made the model worse. This could be because of more aggressive data augmentation, which can make some animals appear too altered.

```
aug = tf.keras.Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.2),
    layers.RandomBrightness(0.3),
    layers.RandomContrast(0.3),
    layers.RandomTranslation(height_factor=0.2, width_factor=0.2),
    layers.RandomZoom(0.2),
    layers.RandomCrop(img_height, img_width),
])

model = Sequential([
    aug,
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.5),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(len(train_ds.class_names), activation='softmax')
])

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

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

epochs = 150

history = model.fit(
    train,
    validation_data=val,
    epochs=epochs,
    callbacks=[early_stopping]
)
```

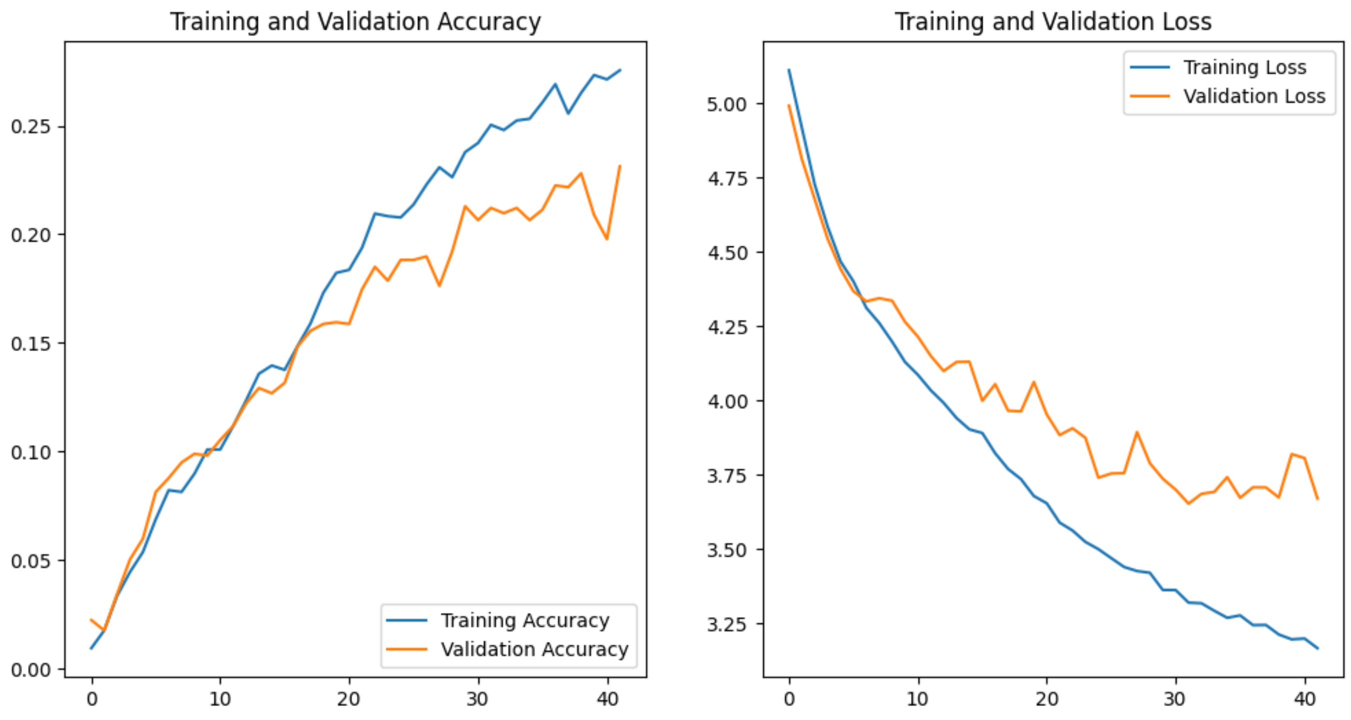




Added Regulizers to Simplify Model

Reduced the number of augmentation layers, as well as the intensity, to maybe help the training accuracy to improve. Also added L2 regularization to add a penalty on any weights on the model that were too large, which could help simplify the model. The validation accuracy of this section is 23.13%.

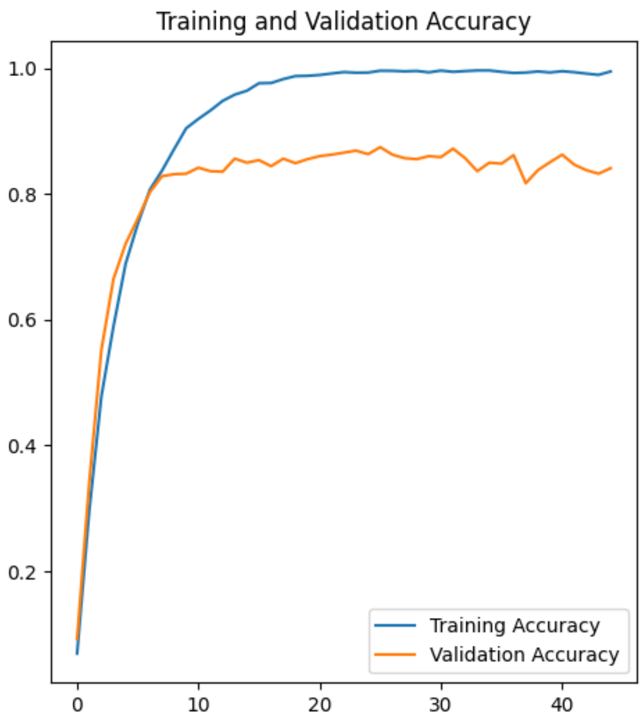
```
from tensorflow.keras import regularizers

aug = tf.keras.Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.2),
    layers.RandomBrightness(0.3),
    layers.RandomContrast(0.3),
])

model = Sequential([
    aug,
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(len(train_ds.class_names), activation='softmax')
])

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

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

epochs = 150

history = model.fit(
    train,
    validation_data=val,
    epochs=epochs,
    callbacks=[early_stopping]
)
```

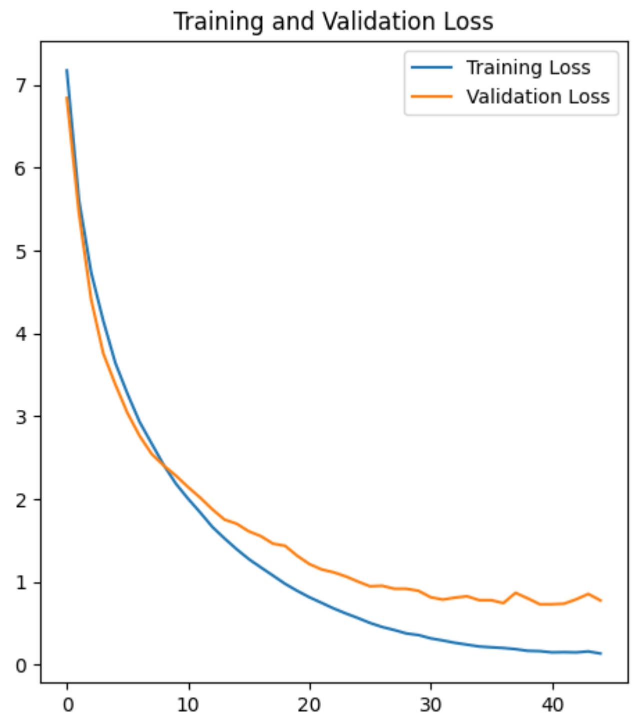


Final Attempt

The most important change in this section was the use of MobileNetV2, which is a pre-trained convolutional neural network that is specialized for image classifications and image feature extraction. MobileNetV2's weights can also be trainable, which allows for the fine tuning to adapt to this data. I also use L2 Regularization to punish large weights to protect against overfitting, and better generalize the data through batch normalization and dropout layers. This new strategy improved the model dramatically, as the validation accuracy is now 87.48% at it's highest, and with a validation loss of 0.7290 at it's lowest.

```python
import tensorflow as tf
from tensorflow.keras import layers, Sequential
from tensorflow.keras.applications import MobileNetV2

img_height = 224
img_width = 224
batch_size = 32

aug = tf.keras.Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.2),
    layers.RandomBrightness(0.2),
    layers.RandomContrast(0.2),
])

data_dir = final_dataset_dir
train_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset='training',
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)

val_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset='validation',
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)

train = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val = val_ds.cache().prefetch(buffer_size=tf.data.AUTOTUNE)

base_model = MobileNetV2(
    input_shape=(img_height, img_width, 3),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = True


model = Sequential([
    aug,
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(len(train_ds.class_names), activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

epochs = 150
history = model.fit(
    train,
    validation_data=val,
    epochs=epochs,
    callbacks=[early_stopping]
)
```


Training and Validation Accuracy
Training Accuracy
Validation Accuracy


Training and Validation Loss
Training Loss
Validation Loss