In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf
from os import listdir
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
from PIL import Image as PImage

import pathlib

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

# For local testing, copy and paste the directory where your data is.
train_path = "/home/jgoh4/card_clf/train/"
data_dir_train = pathlib.Path(train_path).with_suffix('')
val_path = "/home/jgoh4/card_clf/valid/"
data_dir_val = pathlib.Path(val_path).with_suffix('')
test_path = "/home/jgoh4/card_clf/train/"
data_dir_test = pathlib.Path(test_path).with_suffix('')

print(data_dir_train)
```

```
2023-12-07 18:13:54.832410: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operat
ions:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
/home/jgoh4/card_clf/train
```

In [2]:
```
image_count = len(list(data_dir_train.glob('*/*.jpg')))
print("Image Count: ", image_count)
```

```
Image Count:  7624
```

In [3]:
```
# Here we read the .png images into a dataset (non-tabular, presumably) formatted as tensors
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir_train,
  image_size = (224,224),
  batch_size = 32)
```

```
# Here we read the .png images into a dataset (non-tabular, presumably) formatted as tensors
val_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_val,
    image_size = (224,224),
    batch_size = 32
)

# read test images
test_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_test,
    image_size = (224,224),
    batch_size = 32
)

class_names = train_ds.class_names
print(class_names) # same as labels
num_classes = len(class_names)
```

```
Found 7624 files belonging to 53 classes.
2023-12-07 18:14:01.194871: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operat
ions:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-07 18:14:01.819172: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device /job:localhost/re
plica:0/task:0/device:GPU:0 with 78962 MB memory:  -> device: 0, name: NVIDIA A100-SXM4-80GB, pci bus id: 0000:01:00.
0, compute capability: 8.0
Found 265 files belonging to 53 classes.
Found 7624 files belonging to 53 classes.
['ace of clubs', 'ace of diamonds', 'ace of hearts', 'ace of spades', 'eight of clubs', 'eight of diamonds', 'eight o
f hearts', 'eight of spades', 'five of clubs', 'five of diamonds', 'five of hearts', 'five of spades', 'four of club
s', 'four of diamonds', 'four of hearts', 'four of spades', 'jack of clubs', 'jack of diamonds', 'jack of hearts', 'j
ack of spades', 'joker', 'king of clubs', 'king of diamonds', 'king of hearts', 'king of spades', 'nine of clubs', 'n
ine of diamonds', 'nine of hearts', 'nine of spades', 'queen of clubs', 'queen of diamonds', 'queen of hearts', 'quee
n of spades', 'seven of clubs', 'seven of diamonds', 'seven of hearts', 'seven of spades', 'six of clubs', 'six of di
amonds', 'six of hearts', 'six of spades', 'ten of clubs', 'ten of diamonds', 'ten of hearts', 'ten of spades', 'thre
e of clubs', 'three of diamonds', 'three of hearts', 'three of spades', 'two of clubs', 'two of diamonds', 'two of he
arts', 'two of spades']
```

In [4]:

```
# Some kind of optimization - reduces training time
AUTOTUNE = tf.data.AUTOTUNE
```

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

In [5]:
```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
from tensorflow.keras.layers import BatchNormalization
```

Transfer Learning with ResNet50

In [6]:
```
base_model = keras.applications.ResNet50(
    weights="imagenet",  # Load weights pre-trained on ImageNet.
    input_shape=(224, 224, 3),
    include_top=False,
    pooling = max
)  # Do not include the ImageNet classifier at the top.

# Freeze the base_model
base_model.trainable = False

# Data augmentation - necessary for addressing overfitting
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2),
])

# Create new model on top
# inputs = keras.Input(shape=(224, 224, 3))

"""
# Re-scale
scale_layer = keras.layers.Rescaling(scale=1./255, offset=-1)
x = scale_layer(inputs)

x = base_model(x, training = False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)  # Regularize with dropout
outputs = keras.layers.Dense(num_classes)(x)
model = keras.Model(inputs, outputs)
"""
```

```
model = Sequential([
    layers.Rescaling(1./255, input_shape=(224, 224, 3)),
    base_model,
    BatchNormalization(),
    keras.layers.GlobalAveragePooling2D(),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(num_classes, activation = 'softmax') # this line MUST correspond to the number of classes/labe
])
```

In [7]:
```
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy', #SparseCategoricalCrossentropy exclusively for integer-labeled
              metrics=['accuracy'])
```

In [8]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0

 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 batch_normalization (BatchN  (None, 7, 7, 2048)       8192
 ormalization)

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 53)                108597

=================================================================
Total params: 23,704,501
Trainable params: 112,693
Non-trainable params: 23,591,808
_________________________________________________________________
```

In [9]:
```
epochs = 20
res_hist = model.fit(train_ds, epochs=epochs, validation_data=val_ds)
```

```
Epoch 1/20
2023-12-05 01:47:16.343300: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8401
2023-12-05 01:47:17.092985: I tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child process: N
o such file or directory
2023-12-05 01:47:17.150115: I tensorflow/stream_executor/cuda/cuda_blas.cc:1614] TensorFloat-32 will be used for the
matrix multiplication. This will only be logged once.
```

```
239/239 [==============================] - 12s 20ms/step - loss: 3.3001 - accuracy: 0.1891 - val_loss: 3.0539 - val_a
ccuracy: 0.2868
Epoch 2/20
239/239 [==============================] - 4s 15ms/step - loss: 2.6761 - accuracy: 0.3446 - val_loss: 2.4418 - val_ac
curacy: 0.3887
Epoch 3/20
239/239 [==============================] - 4s 15ms/step - loss: 2.3671 - accuracy: 0.4234 - val_loss: 2.1332 - val_ac
curacy: 0.4792
Epoch 4/20
239/239 [==============================] - 4s 15ms/step - loss: 2.1500 - accuracy: 0.4757 - val_loss: 2.0235 - val_ac
curacy: 0.5132
Epoch 5/20
239/239 [==============================] - 4s 15ms/step - loss: 1.9905 - accuracy: 0.5153 - val_loss: 1.9308 - val_ac
curacy: 0.5321
Epoch 6/20
239/239 [==============================] - 4s 15ms/step - loss: 1.8640 - accuracy: 0.5460 - val_loss: 1.8586 - val_ac
curacy: 0.5321
Epoch 7/20
239/239 [==============================] - 4s 15ms/step - loss: 1.7535 - accuracy: 0.5727 - val_loss: 1.8592 - val_ac
curacy: 0.5283
Epoch 8/20
239/239 [==============================] - 4s 15ms/step - loss: 1.6679 - accuracy: 0.5934 - val_loss: 1.8003 - val_ac
curacy: 0.5547
Epoch 9/20
239/239 [==============================] - 4s 15ms/step - loss: 1.5863 - accuracy: 0.6132 - val_loss: 1.7673 - val_ac
curacy: 0.5623
Epoch 10/20
239/239 [==============================] - 4s 17ms/step - loss: 1.5185 - accuracy: 0.6295 - val_loss: 1.7566 - val_ac
curacy: 0.5585
Epoch 11/20
239/239 [==============================] - 4s 18ms/step - loss: 1.4538 - accuracy: 0.6445 - val_loss: 1.7385 - val_ac
curacy: 0.5698
Epoch 12/20
239/239 [==============================] - 5s 19ms/step - loss: 1.4021 - accuracy: 0.6550 - val_loss: 1.7762 - val_ac
curacy: 0.5736
Epoch 13/20
239/239 [==============================] - 4s 18ms/step - loss: 1.3417 - accuracy: 0.6735 - val_loss: 1.7169 - val_ac
curacy: 0.5660
Epoch 14/20
239/239 [==============================] - 4s 17ms/step - loss: 1.2961 - accuracy: 0.6777 - val_loss: 1.7638 - val_ac
curacy: 0.5811
Epoch 15/20
```

```
239/239 [==============================] - 4s 18ms/step - loss: 1.2544 - accuracy: 0.6929 - val_loss: 1.7911 - val_ac
curacy: 0.5925
Epoch 16/20
239/239 [==============================] - 4s 18ms/step - loss: 1.2123 - accuracy: 0.7015 - val_loss: 1.7894 - val_ac
curacy: 0.5660
Epoch 17/20
239/239 [==============================] - 4s 19ms/step - loss: 1.1746 - accuracy: 0.7108 - val_loss: 1.7578 - val_ac
curacy: 0.5698
Epoch 18/20
239/239 [==============================] - 4s 18ms/step - loss: 1.1473 - accuracy: 0.7162 - val_loss: 1.7692 - val_ac
curacy: 0.6000
Epoch 19/20
239/239 [==============================] - 4s 18ms/step - loss: 1.1151 - accuracy: 0.7231 - val_loss: 1.8363 - val_ac
curacy: 0.5698
Epoch 20/20
239/239 [==============================] - 4s 18ms/step - loss: 1.0815 - accuracy: 0.7280 - val_loss: 1.7745 - val_ac
curacy: 0.5962
```

In [10]:

```python
# Without finetuning, overfitting with increaseing epochs. Peak at 60%
# Fine-tune model. Unfreeze base model, train end-to-end with low learning rate
base_model.trainable = True
model.summary()

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss='sparse_categorical_crossentropy', #SparseCategoricalCrossentropy exclusively for integer-labeled
    metrics=['accuracy'])

epochs = 20
res_hist = model.fit(train_ds, epochs=epochs, validation_data=val_ds)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0

 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 batch_normalization (BatchN  (None, 7, 7, 2048)       8192
 ormalization)

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 53)                108597

=================================================================
Total params: 23,704,501
Trainable params: 23,647,285
Non-trainable params: 57,216
_________________________________________________________________
Epoch 1/20
239/239 [==============================] - 19s 61ms/step - loss: 8.9092 - accuracy: 0.1170 - val_loss: 15.4119 - val_
accuracy: 0.0340
Epoch 2/20
239/239 [==============================] - 14s 57ms/step - loss: 3.7628 - accuracy: 0.3808 - val_loss: 12.1304 - val_
accuracy: 0.0755
Epoch 3/20
239/239 [==============================] - 14s 57ms/step - loss: 1.9948 - accuracy: 0.5876 - val_loss: 6.5839 - val_a
ccuracy: 0.2189
Epoch 4/20
239/239 [==============================] - 14s 59ms/step - loss: 1.0996 - accuracy: 0.7263 - val_loss: 3.2015 - val_a
ccuracy: 0.4453
Epoch 5/20
239/239 [==============================] - 13s 56ms/step - loss: 0.5893 - accuracy: 0.8374 - val_loss: 2.3061 - val_a
ccuracy: 0.5811
Epoch 6/20
239/239 [==============================] - 13s 56ms/step - loss: 0.3291 - accuracy: 0.9035 - val_loss: 2.2252 - val_a
```

```
ccuracy: 0.5849
Epoch 7/20
239/239 [==============================] - 14s 57ms/step - loss: 0.1778 - accuracy: 0.9446 - val_loss: 2.1867 - val_a
ccuracy: 0.6226
Epoch 8/20
239/239 [==============================] - 14s 57ms/step - loss: 0.1183 - accuracy: 0.9634 - val_loss: 2.0859 - val_a
ccuracy: 0.6189
Epoch 9/20
239/239 [==============================] - 13s 56ms/step - loss: 0.0816 - accuracy: 0.9755 - val_loss: 2.0337 - val_a
ccuracy: 0.6377
Epoch 10/20
239/239 [==============================] - 14s 59ms/step - loss: 0.0551 - accuracy: 0.9860 - val_loss: 2.0388 - val_a
ccuracy: 0.6189
Epoch 11/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0435 - accuracy: 0.9881 - val_loss: 2.0104 - val_a
ccuracy: 0.6302
Epoch 12/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0329 - accuracy: 0.9924 - val_loss: 2.0378 - val_a
ccuracy: 0.6264
Epoch 13/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0271 - accuracy: 0.9937 - val_loss: 2.0181 - val_a
ccuracy: 0.6453
Epoch 14/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0230 - accuracy: 0.9938 - val_loss: 1.9652 - val_a
ccuracy: 0.6377
Epoch 15/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0183 - accuracy: 0.9967 - val_loss: 1.9041 - val_a
ccuracy: 0.6491
Epoch 16/20
239/239 [==============================] - 14s 59ms/step - loss: 0.0174 - accuracy: 0.9966 - val_loss: 1.9843 - val_a
ccuracy: 0.6717
Epoch 17/20
239/239 [==============================] - 14s 59ms/step - loss: 0.0152 - accuracy: 0.9969 - val_loss: 1.9441 - val_a
ccuracy: 0.6604
Epoch 18/20
239/239 [==============================] - 14s 59ms/step - loss: 0.0143 - accuracy: 0.9972 - val_loss: 1.9522 - val_a
ccuracy: 0.6642
Epoch 19/20
239/239 [==============================] - 14s 57ms/step - loss: 0.0115 - accuracy: 0.9978 - val_loss: 2.0128 - val_a
ccuracy: 0.6491
Epoch 20/20
```

```
239/239 [==============================] - 14s 57ms/step - loss: 0.0094 - accuracy: 0.9984 - val_loss: 1.9493 - val_a
ccuracy: 0.6830
```

In [11]:
```python
# save 'model history' as a dictionary
import pickle
with open('/trainHistoryDict', 'wb') as file_pi:
    pickle.dump(res_hist.history, file_pi)
```

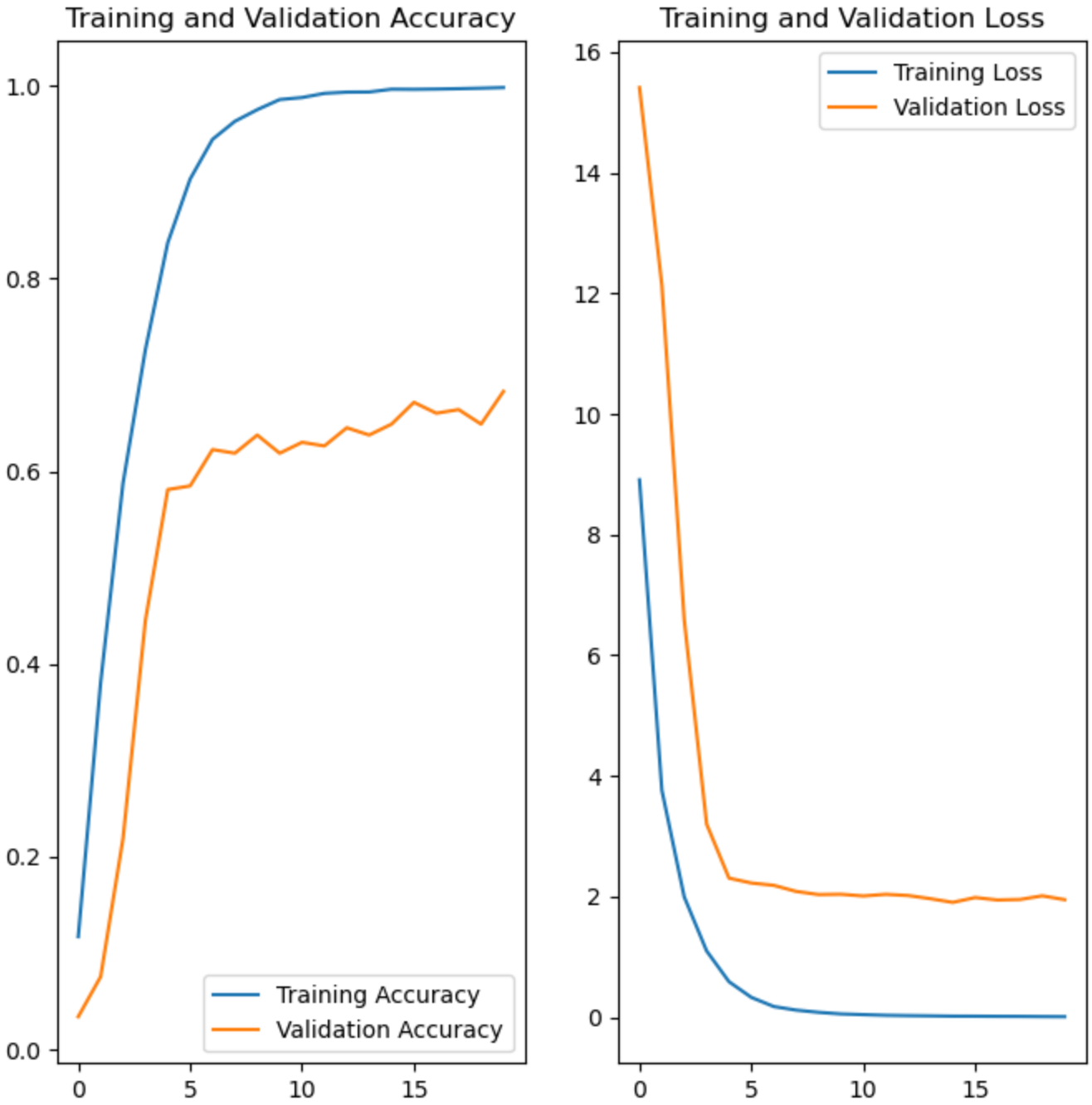
```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
Cell In [11], line 3
      1 # save 'model history' as a dictionary
      2 import pickle
----> 3 with open('/trainHistoryDict', 'wb') as file_pi:
      4     pickle.dump(res_hist.history, file_pi)

File /packages/envs/tensorflow-gpu-2.10.0/lib/python3.8/site-packages/IPython/core/interactiveshell.py:282, in _modif
ied_open(file, *args, **kwargs)
    275 if file in {0, 1, 2}:
    276     raise ValueError(
    277         f"IPython won't let you open fd={file} by default "
    278         "as it is likely to crash IPython. If you know what you are doing, "
    279         "you can use builtins' open."
    280     )
--> 282 return io_open(file, *args, **kwargs)

PermissionError: [Errno 13] Permission denied: '/trainHistoryDict'
```

In [ ]:
```python
# load history. for plotting loss/accuracy
with open('/trainHistoryDict', "rb") as file_pi:
    res_history_load = pickle.load(file_pi)
```

In [12]:
```python
acc = res_hist.history['accuracy']
val_acc = res_hist.history['val_accuracy']
print(len(acc))
loss = res_hist.history['loss']
val_loss = res_hist.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
```

```
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
# With fine-tuning, our validation accuracy plataues
```

```
20
```


Training and Validation Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
Training Accuracy
Validation Accuracy
Training and Validation Loss
16
14
12
10
8
6
4
2
0
0
5
10
15
Training Loss
Validation Loss

In [ ]:
```
# with fine-tuning, we see diminishing returns in validation loss. Other methods must be utilized to address this ove
# Consider hypermodel for ResNet
```

HyperModel, based on GridSearch

In [6]:
```
import keras_tuner as kt
from keras_tuner.applications import HyperResNet
```

In [7]:
```
tuner_gs = kt.GridSearch(HyperResNet(include_top = False, input_shape=(224, 224, 3), classes=num_classes),
                     objective='val_accuracy',
                     max_trials=20, max_consecutive_failed_trials = 5)
```

```
Reloading Tuner from ./untitled_project/tuner0.json
```

In [8]:
```
# Create a callback to stop training early after reaching a certain value for the validation loss. If no improvement
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
```

In [19]:
```
# Run hyperparamater search!
# Same arguments as in model.fit
tuner_gs.search(train_ds, epochs=50, validation_data = val_ds, callbacks=[stop_early])
# Get the optimal hyperparameters
best_hps_hyp_gs=tuner_gs.get_best_hyperparameters(num_trials=1)[0]

print(f"""
The hyperparameter search (using HyperResNet) is complete. {best_hps_hyp_gs.values}
""")
```

```
The hyperparameter search (using HyperResNet) is complete. {'version': 'v2', 'conv3_depth': 4, 'conv4_depth': 6, 'poo
ling': 'avg', 'optimizer': 'adam', 'learning_rate': 0.001}
```

In [20]:
```
# Build the model with the optimal hyperparameters and train it on the data for 50 epochs
model = tuner_gs.hypermodel.build(best_hps_hyp_gs)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=best_hps_hyp_gs.get('learning_rate')),
                loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                metrics=['accuracy'])
history_hp_hyp_gs = model.fit(train_ds, epochs=50, validation_data=val_ds)


val_acc_per_epoch = history_hp_hyp_gs.history['val_accuracy']
```

```
best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1
print('Best epoch: %d' % (best_epoch,))
```

```
Epoch 1/50
2023-12-07 18:24:41.556982: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8401
2023-12-07 18:24:47.514312: I tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child process: N
o such file or directory
2023-12-07 18:24:48.093958: I tensorflow/stream_executor/cuda/cuda_blas.cc:1614] TensorFloat-32 will be used for the
matrix multiplication. This will only be logged once.
```

```
239/239 [==============================] - 27s 48ms/step - loss: 6.2392 - accuracy: 0.1272 - val_loss: 5.8682 - val_a
ccuracy: 0.0981
Epoch 2/50
239/239 [==============================] - 10s 42ms/step - loss: 5.0818 - accuracy: 0.2315 - val_loss: 5.7359 - val_a
ccuracy: 0.0830
Epoch 3/50
239/239 [==============================] - 10s 41ms/step - loss: 4.3152 - accuracy: 0.2933 - val_loss: 5.2067 - val_a
ccuracy: 0.1660
Epoch 4/50
239/239 [==============================] - 10s 42ms/step - loss: 3.7167 - accuracy: 0.3615 - val_loss: 3.2226 - val_a
ccuracy: 0.2604
Epoch 5/50
239/239 [==============================] - 10s 41ms/step - loss: 3.1697 - accuracy: 0.4334 - val_loss: 3.1604 - val_a
ccuracy: 0.4000
Epoch 6/50
239/239 [==============================] - 10s 42ms/step - loss: 2.7476 - accuracy: 0.4807 - val_loss: 2.5376 - val_a
ccuracy: 0.4679
Epoch 7/50
239/239 [==============================] - 10s 42ms/step - loss: 2.3854 - accuracy: 0.5414 - val_loss: 2.4351 - val_a
ccuracy: 0.4830
Epoch 8/50
239/239 [==============================] - 10s 42ms/step - loss: 1.9948 - accuracy: 0.6325 - val_loss: 1.2404 - val_a
ccuracy: 0.7585
Epoch 9/50
239/239 [==============================] - 10s 42ms/step - loss: 1.6609 - accuracy: 0.6910 - val_loss: 1.0199 - val_a
ccuracy: 0.8038
Epoch 10/50
239/239 [==============================] - 10s 42ms/step - loss: 1.3734 - accuracy: 0.7395 - val_loss: 1.5226 - val_a
ccuracy: 0.7057
Epoch 11/50
239/239 [==============================] - 10s 42ms/step - loss: 1.1161 - accuracy: 0.7911 - val_loss: 0.7631 - val_a
ccuracy: 0.8717
Epoch 12/50
239/239 [==============================] - 10s 42ms/step - loss: 0.8846 - accuracy: 0.8332 - val_loss: 0.6696 - val_a
ccuracy: 0.8415
Epoch 13/50
239/239 [==============================] - 10s 42ms/step - loss: 0.6902 - accuracy: 0.8762 - val_loss: 0.7393 - val_a
ccuracy: 0.8453
Epoch 14/50
239/239 [==============================] - 10s 42ms/step - loss: 0.5585 - accuracy: 0.9006 - val_loss: 0.8756 - val_a
ccuracy: 0.8226
Epoch 15/50
```

```
239/239 [==============================] - 10s 42ms/step - loss: 0.4790 - accuracy: 0.9154 - val_loss: 0.5746 - val_a
ccuracy: 0.8906
Epoch 16/50
239/239 [==============================] - 10s 42ms/step - loss: 0.3864 - accuracy: 0.9315 - val_loss: 0.8987 - val_a
ccuracy: 0.8226
Epoch 17/50
239/239 [==============================] - 10s 42ms/step - loss: 0.3086 - accuracy: 0.9523 - val_loss: 0.5815 - val_a
ccuracy: 0.8830
Epoch 18/50
239/239 [==============================] - 10s 42ms/step - loss: 0.2418 - accuracy: 0.9669 - val_loss: 0.5906 - val_a
ccuracy: 0.8868
Epoch 19/50
239/239 [==============================] - 10s 41ms/step - loss: 0.2100 - accuracy: 0.9708 - val_loss: 0.5732 - val_a
ccuracy: 0.8792
Epoch 20/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1874 - accuracy: 0.9784 - val_loss: 0.4453 - val_a
ccuracy: 0.8943
Epoch 21/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1854 - accuracy: 0.9738 - val_loss: 0.5979 - val_a
ccuracy: 0.8906
Epoch 22/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1936 - accuracy: 0.9704 - val_loss: 0.6546 - val_a
ccuracy: 0.8906
Epoch 23/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1599 - accuracy: 0.9774 - val_loss: 0.4620 - val_a
ccuracy: 0.9094
Epoch 24/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1098 - accuracy: 0.9891 - val_loss: 0.5732 - val_a
ccuracy: 0.8792
Epoch 25/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1165 - accuracy: 0.9865 - val_loss: 0.7066 - val_a
ccuracy: 0.8792
Epoch 26/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1533 - accuracy: 0.9756 - val_loss: 0.7112 - val_a
ccuracy: 0.8491
Epoch 27/50
239/239 [==============================] - 10s 42ms/step - loss: 0.1594 - accuracy: 0.9727 - val_loss: 0.6624 - val_a
ccuracy: 0.8755
Epoch 28/50
239/239 [==============================] - 10s 41ms/step - loss: 0.1377 - accuracy: 0.9768 - val_loss: 0.4440 - val_a
ccuracy: 0.9245
Epoch 29/50
```

```
239/239 [==============================] - 10s 42ms/step - loss: 0.0707 - accuracy: 0.9941 - val_loss: 0.3475 - val_a
ccuracy: 0.9321
Epoch 30/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0502 - accuracy: 0.9967 - val_loss: 0.3153 - val_a
ccuracy: 0.9321
Epoch 31/50
239/239 [==============================] - 10s 41ms/step - loss: 0.0364 - accuracy: 0.9987 - val_loss: 0.3194 - val_a
ccuracy: 0.9321
Epoch 32/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0293 - accuracy: 0.9990 - val_loss: 0.2879 - val_a
ccuracy: 0.9472
Epoch 33/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0222 - accuracy: 1.0000 - val_loss: 0.2986 - val_a
ccuracy: 0.9434
Epoch 34/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0190 - accuracy: 1.0000 - val_loss: 0.3312 - val_a
ccuracy: 0.9472
Epoch 35/50
239/239 [==============================] - 10s 41ms/step - loss: 0.0169 - accuracy: 0.9999 - val_loss: 0.2954 - val_a
ccuracy: 0.9358
Epoch 36/50
239/239 [==============================] - 10s 41ms/step - loss: 0.0146 - accuracy: 1.0000 - val_loss: 0.3126 - val_a
ccuracy: 0.9358
Epoch 37/50
239/239 [==============================] - 10s 41ms/step - loss: 0.0132 - accuracy: 1.0000 - val_loss: 0.2989 - val_a
ccuracy: 0.9396
Epoch 38/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0121 - accuracy: 1.0000 - val_loss: 0.3160 - val_a
ccuracy: 0.9358
Epoch 39/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0114 - accuracy: 1.0000 - val_loss: 0.3139 - val_a
ccuracy: 0.9472
Epoch 40/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0107 - accuracy: 1.0000 - val_loss: 0.3236 - val_a
ccuracy: 0.9434
Epoch 41/50
239/239 [==============================] - 10s 42ms/step - loss: 0.5108 - accuracy: 0.8810 - val_loss: 1.6088 - val_a
ccuracy: 0.6566
Epoch 42/50
239/239 [==============================] - 10s 42ms/step - loss: 0.3695 - accuracy: 0.9033 - val_loss: 0.5469 - val_a
ccuracy: 0.8981
Epoch 43/50
```

```
239/239 [==============================] - 10s 42ms/step - loss: 0.1007 - accuracy: 0.9814 - val_loss: 0.4272 - val_a
ccuracy: 0.8981
Epoch 44/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0440 - accuracy: 0.9948 - val_loss: 0.3856 - val_a
ccuracy: 0.9094
Epoch 45/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0272 - accuracy: 0.9978 - val_loss: 0.2816 - val_a
ccuracy: 0.9396
Epoch 46/50
239/239 [==============================] - 10s 41ms/step - loss: 0.0201 - accuracy: 0.9988 - val_loss: 0.3038 - val_a
ccuracy: 0.9434
Epoch 47/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0128 - accuracy: 0.9999 - val_loss: 0.3224 - val_a
ccuracy: 0.9434
Epoch 48/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0100 - accuracy: 0.9999 - val_loss: 0.2939 - val_a
ccuracy: 0.9396
Epoch 49/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0080 - accuracy: 1.0000 - val_loss: 0.3119 - val_a
ccuracy: 0.9396
Epoch 50/50
239/239 [==============================] - 10s 42ms/step - loss: 0.0071 - accuracy: 1.0000 - val_loss: 0.3115 - val_a
ccuracy: 0.9396
Best epoch: 32
```

In [23]:

```python
# Re-instantiate the hypermodel and train it with the optimal number of epochs from above.
hypermodel = tuner_gs.hypermodel.build(best_hps_hyp_gs)
hypermodel.compile(optimizer=keras.optimizers.Adam(learning_rate=best_hps_hyp_gs.get('learning_rate')),
                   loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])
# Retrain the model
final_hist_hyp_gs = hypermodel.fit(train_ds, epochs=best_epoch, validation_data = val_ds)
```

```
Epoch 1/32
239/239 [==============================] - 13s 43ms/step - loss: 6.2046 - accuracy: 0.1403 - val_loss: 13.2003 - val_
accuracy: 0.0302
Epoch 2/32
239/239 [==============================] - 10s 41ms/step - loss: 5.0901 - accuracy: 0.2281 - val_loss: 6.7965 - val_a
ccuracy: 0.0491
Epoch 3/32
239/239 [==============================] - 10s 41ms/step - loss: 4.2955 - accuracy: 0.2926 - val_loss: 3.2618 - val_a
ccuracy: 0.3585
Epoch 4/32
239/239 [==============================] - 10s 41ms/step - loss: 3.7139 - accuracy: 0.3498 - val_loss: 3.5947 - val_a
ccuracy: 0.2717
Epoch 5/32
239/239 [==============================] - 10s 41ms/step - loss: 3.1901 - accuracy: 0.4218 - val_loss: 4.2741 - val_a
ccuracy: 0.2113
Epoch 6/32
239/239 [==============================] - 10s 41ms/step - loss: 2.7229 - accuracy: 0.4887 - val_loss: 1.8999 - val_a
ccuracy: 0.5660
Epoch 7/32
239/239 [==============================] - 10s 41ms/step - loss: 2.3089 - accuracy: 0.5771 - val_loss: 2.0587 - val_a
ccuracy: 0.5925
Epoch 8/32
239/239 [==============================] - 10s 41ms/step - loss: 1.9034 - accuracy: 0.6612 - val_loss: 1.2756 - val_a
ccuracy: 0.6943
Epoch 9/32
239/239 [==============================] - 10s 41ms/step - loss: 1.5828 - accuracy: 0.7065 - val_loss: 1.9173 - val_a
ccuracy: 0.5887
Epoch 10/32
239/239 [==============================] - 10s 41ms/step - loss: 1.3335 - accuracy: 0.7425 - val_loss: 1.4460 - val_a
ccuracy: 0.7094
Epoch 11/32
239/239 [==============================] - 10s 41ms/step - loss: 1.0974 - accuracy: 0.7871 - val_loss: 0.8477 - val_a
ccuracy: 0.8113
Epoch 12/32
239/239 [==============================] - 10s 41ms/step - loss: 0.8849 - accuracy: 0.8279 - val_loss: 0.7728 - val_a
ccuracy: 0.8528
Epoch 13/32
239/239 [==============================] - 10s 41ms/step - loss: 0.6963 - accuracy: 0.8699 - val_loss: 0.9548 - val_a
ccuracy: 0.8189
Epoch 14/32
239/239 [==============================] - 10s 41ms/step - loss: 0.5648 - accuracy: 0.8980 - val_loss: 0.9282 - val_a
ccuracy: 0.8189
```

```
Epoch 15/32
239/239 [==============================] - 10s 41ms/step - loss: 0.4433 - accuracy: 0.9264 - val_loss: 0.4958 - val_a
ccuracy: 0.9132
Epoch 16/32
239/239 [==============================] - 10s 41ms/step - loss: 0.3573 - accuracy: 0.9446 - val_loss: 0.6141 - val_a
ccuracy: 0.8528
Epoch 17/32
239/239 [==============================] - 10s 41ms/step - loss: 0.2988 - accuracy: 0.9566 - val_loss: 0.5357 - val_a
ccuracy: 0.8906
Epoch 18/32
239/239 [==============================] - 10s 41ms/step - loss: 0.2585 - accuracy: 0.9645 - val_loss: 0.6992 - val_a
ccuracy: 0.8642
Epoch 19/32
239/239 [==============================] - 10s 41ms/step - loss: 0.2265 - accuracy: 0.9668 - val_loss: 0.6887 - val_a
ccuracy: 0.8604
Epoch 20/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1928 - accuracy: 0.9765 - val_loss: 0.5875 - val_a
ccuracy: 0.8792
Epoch 21/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1974 - accuracy: 0.9718 - val_loss: 0.4862 - val_a
ccuracy: 0.9019
Epoch 22/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1581 - accuracy: 0.9816 - val_loss: 0.4531 - val_a
ccuracy: 0.8981
Epoch 23/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1588 - accuracy: 0.9772 - val_loss: 0.8614 - val_a
ccuracy: 0.8189
Epoch 24/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1443 - accuracy: 0.9814 - val_loss: 0.5677 - val_a
ccuracy: 0.8755
Epoch 25/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1525 - accuracy: 0.9755 - val_loss: 0.4463 - val_a
ccuracy: 0.9094
Epoch 26/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1299 - accuracy: 0.9833 - val_loss: 0.4520 - val_a
ccuracy: 0.8943
Epoch 27/32
239/239 [==============================] - 10s 41ms/step - loss: 0.0893 - accuracy: 0.9904 - val_loss: 0.4161 - val_a
ccuracy: 0.9132
Epoch 28/32
239/239 [==============================] - 10s 41ms/step - loss: 0.0894 - accuracy: 0.9891 - val_loss: 0.3617 - val_a
ccuracy: 0.9358
```

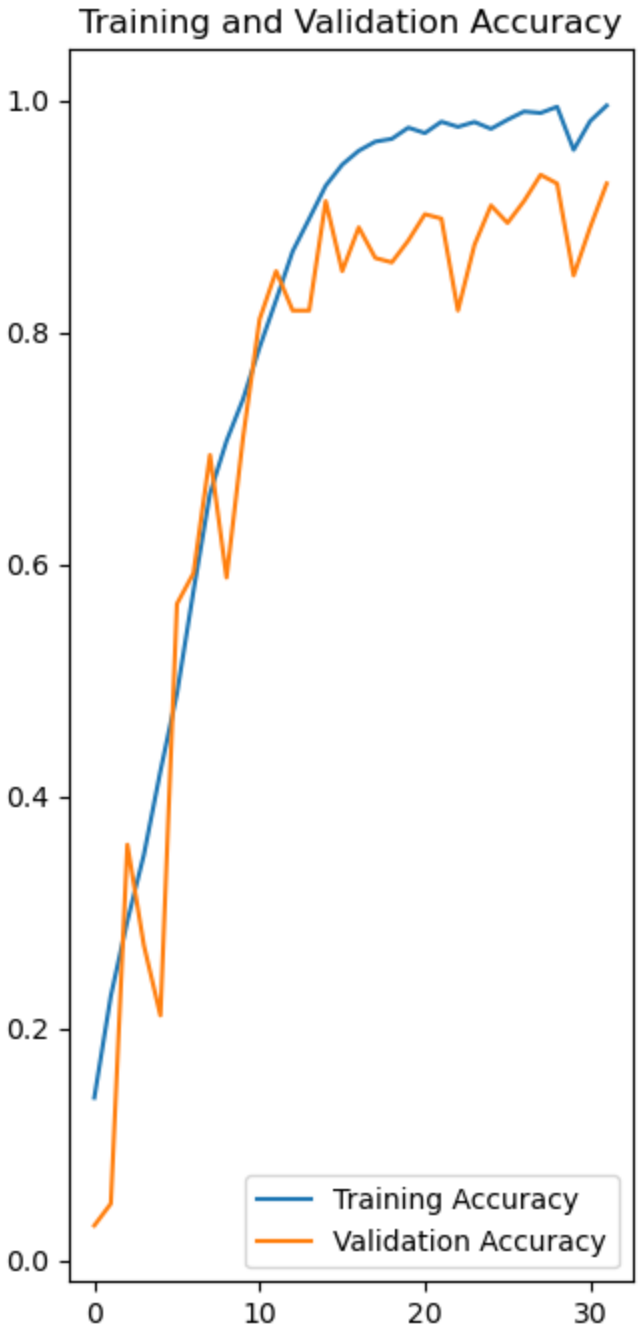
```
Epoch 29/32
239/239 [==============================] - 10s 41ms/step - loss: 0.0660 - accuracy: 0.9945 - val_loss: 0.4139 - val_a
ccuracy: 0.9283
Epoch 30/32
239/239 [==============================] - 10s 41ms/step - loss: 0.2044 - accuracy: 0.9574 - val_loss: 0.8025 - val_a
ccuracy: 0.8491
Epoch 31/32
239/239 [==============================] - 10s 41ms/step - loss: 0.1121 - accuracy: 0.9822 - val_loss: 0.5978 - val_a
ccuracy: 0.8906
Epoch 32/32
239/239 [==============================] - 10s 41ms/step - loss: 0.0564 - accuracy: 0.9955 - val_loss: 0.3384 - val_a
ccuracy: 0.9283
32
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [23], line 15
     12 loss = final_hist_hyp_gs.history['loss']
     13 val_loss = final_hist_hyp_gs.history['val_loss']
---> 15 epochs_range = range(epochs)
     17 plt.figure(figsize=(8, 8))
     18 plt.subplot(1, 2, 1)

NameError: name 'epochs' is not defined
```

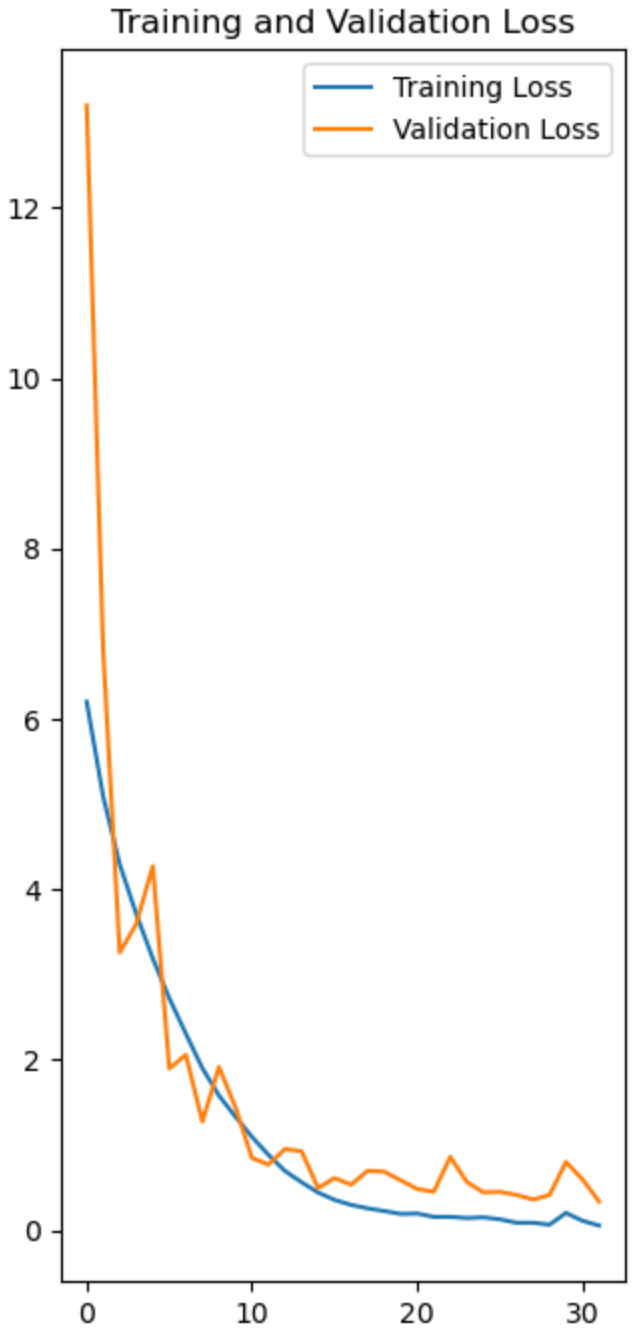
In [24]:

```python
acc = final_hist_hyp_gs.history['accuracy']
val_acc = final_hist_hyp_gs.history['val_accuracy']
print(len(acc))
loss = final_hist_hyp_gs.history['loss']
val_loss = final_hist_hyp_gs.history['val_loss']

epochs_range = range(best_epoch)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
```

```
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

32


Training and Validation Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
Training Accuracy
Validation Accuracy


Training and Validation Loss
12
10
8
6
4
2
0
0
10
20
30
Training Loss
Validation Loss

In [25]:
```
eval_result = hypermodel.evaluate(test_ds)
print("[test loss, test accuracy]:", eval_result)
```

```
239/239 [==============================] - 4s 17ms/step - loss: 0.0527 - accuracy: 0.9871
[test loss, test accuracy]: [0.05265914276242256, 0.9871458411216736]
```

HyperModel, based on RandomSearch

In [50]:
```
tuner_rs = kt.RandomSearch(HyperResNet(include_top = False, input_shape=(224, 224, 3), classes=num_classes),
                     objective='val_accuracy',
                     max_trials=20, max_consecutive_failed_trials = 200, max_retries_per_trial = 10) #force trials to
```

```
Reloading Tuner from ./untitled_project/tuner0.json
```

In [51]:
```
# Create a callback to stop training early after reaching a certain value for the validation loss. If no improvement
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
```

In [52]:
```
# Run hyperparamater search!
# Same arguments as in model.fit

tuner_rs.search(train_ds, epochs=50, validation_data = val_ds, callbacks=[stop_early])
# Get the optimal hyperparameters
best_hps_hyp_rs=tuner_rs.get_best_hyperparameters(num_trials=1)[0]

print(f"""
The hyperparameter search (using HyperResNet) is complete. {best_hps_hyp_rs.values}
""")
```

```
Trial 20 Complete [00h 00m 01s]

Best val_accuracy So Far: None
Total elapsed time: 04h 53m 42s

The hyperparameter search (using HyperResNet) is complete. {'version': 'v1', 'conv3_depth': 4, 'conv4_depth': 36, 'po
oling': 'avg', 'optimizer': 'sgd', 'learning_rate': 0.1}
```

In [53]:
```
# Build the model with the optimal hyperparameters and train it on the data for 50 epochs
# We search for the best epoch, with the best measures.
model = tuner_rs.hypermodel.build(best_hps_hyp_rs)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=best_hps_hyp_rs.get('learning_rate')),
```

```
                loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                metrics=['accuracy'])
history_hp_hyp_rs = model.fit(train_ds, epochs=50, validation_data=val_ds)


val_acc_per_epoch = history_hp_hyp_rs.history['val_accuracy']
best_val_acc = max(val_acc_per_epoch)
best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1
print('Best epoch: %d' % (best_epoch,))
```

```
Epoch 1/50
239/239 [==============================] - 34s 105ms/step - loss: 4.6444 - accuracy: 0.0501 - val_loss: 80.3757 - val
_accuracy: 0.0000e+00
Epoch 2/50
239/239 [==============================] - 23s 96ms/step - loss: 3.3555 - accuracy: 0.1762 - val_loss: 6.8624 - val_a
ccuracy: 0.1057
Epoch 3/50
239/239 [==============================] - 24s 99ms/step - loss: 2.7122 - accuracy: 0.2836 - val_loss: 4.9408 - val_a
ccuracy: 0.1358
Epoch 4/50
239/239 [==============================] - 24s 101ms/step - loss: 2.4266 - accuracy: 0.3717 - val_loss: 12.7961 - val
_accuracy: 0.1170
Epoch 5/50
239/239 [==============================] - 24s 101ms/step - loss: 2.1455 - accuracy: 0.4620 - val_loss: 7.3751 - val_
accuracy: 0.1019
Epoch 6/50
239/239 [==============================] - 24s 100ms/step - loss: 1.8681 - accuracy: 0.5518 - val_loss: 6.1565 - val_
accuracy: 0.3094
Epoch 7/50
239/239 [==============================] - 24s 100ms/step - loss: 1.6756 - accuracy: 0.5989 - val_loss: 2.1631 - val_
accuracy: 0.5472
Epoch 8/50
239/239 [==============================] - 24s 101ms/step - loss: 1.4951 - accuracy: 0.6351 - val_loss: 5.0273 - val_
accuracy: 0.2830
Epoch 9/50
239/239 [==============================] - 24s 101ms/step - loss: 1.3649 - accuracy: 0.6742 - val_loss: 2.0160 - val_
accuracy: 0.5811
Epoch 10/50
239/239 [==============================] - 24s 99ms/step - loss: 1.2349 - accuracy: 0.7041 - val_loss: 2.4573 - val_a
ccuracy: 0.5849
Epoch 11/50
239/239 [==============================] - 23s 96ms/step - loss: 1.1257 - accuracy: 0.7299 - val_loss: 1.9958 - val_a
ccuracy: 0.6566
Epoch 12/50
239/239 [==============================] - 23s 96ms/step - loss: 1.0185 - accuracy: 0.7606 - val_loss: 1.1720 - val_a
ccuracy: 0.7170
Epoch 13/50
239/239 [==============================] - 23s 96ms/step - loss: 0.9518 - accuracy: 0.7715 - val_loss: 2.7374 - val_a
ccuracy: 0.5547
Epoch 14/50
239/239 [==============================] - 23s 97ms/step - loss: 0.8516 - accuracy: 0.7968 - val_loss: 3.9188 - val_a
ccuracy: 0.4113
```

```
Epoch 15/50
239/239 [==============================] - 23s 96ms/step - loss: 0.6993 - accuracy: 0.8207 - val_loss: 1.6296 - val_a
ccuracy: 0.6528
Epoch 16/50
239/239 [==============================] - 23s 96ms/step - loss: 0.5515 - accuracy: 0.8497 - val_loss: 0.8380 - val_a
ccuracy: 0.7623
Epoch 17/50
239/239 [==============================] - 23s 97ms/step - loss: 0.4425 - accuracy: 0.8696 - val_loss: 2.6282 - val_a
ccuracy: 0.5585
Epoch 18/50
239/239 [==============================] - 23s 97ms/step - loss: 0.3600 - accuracy: 0.8935 - val_loss: 1.4561 - val_a
ccuracy: 0.6830
Epoch 19/50
239/239 [==============================] - 23s 97ms/step - loss: 0.3292 - accuracy: 0.8970 - val_loss: 1.0045 - val_a
ccuracy: 0.7849
Epoch 20/50
239/239 [==============================] - 23s 97ms/step - loss: 0.2865 - accuracy: 0.9141 - val_loss: 2.1419 - val_a
ccuracy: 0.6264
Epoch 21/50
239/239 [==============================] - 23s 96ms/step - loss: 0.2408 - accuracy: 0.9250 - val_loss: 0.8964 - val_a
ccuracy: 0.8151
Epoch 22/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1865 - accuracy: 0.9465 - val_loss: 1.8391 - val_a
ccuracy: 0.6830
Epoch 23/50
239/239 [==============================] - 23s 96ms/step - loss: 0.2286 - accuracy: 0.9277 - val_loss: 2.1349 - val_a
ccuracy: 0.6302
Epoch 24/50
239/239 [==============================] - 23s 97ms/step - loss: 0.2029 - accuracy: 0.9410 - val_loss: 0.9733 - val_a
ccuracy: 0.7660
Epoch 25/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1660 - accuracy: 0.9482 - val_loss: 1.2359 - val_a
ccuracy: 0.6906
Epoch 26/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1746 - accuracy: 0.9465 - val_loss: 1.4508 - val_a
ccuracy: 0.7208
Epoch 27/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1603 - accuracy: 0.9499 - val_loss: 1.0852 - val_a
ccuracy: 0.7849
Epoch 28/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0849 - accuracy: 0.9742 - val_loss: 0.3803 - val_a
ccuracy: 0.9057
```

```
Epoch 29/50
239/239 [==============================] - 24s 100ms/step - loss: 0.1674 - accuracy: 0.9461 - val_loss: 2.1947 - val_
accuracy: 0.6604
Epoch 30/50
239/239 [==============================] - 24s 101ms/step - loss: 0.1867 - accuracy: 0.9381 - val_loss: 1.2219 - val_
accuracy: 0.7698
Epoch 31/50
239/239 [==============================] - 24s 101ms/step - loss: 0.1238 - accuracy: 0.9621 - val_loss: 1.4223 - val_
accuracy: 0.7358
Epoch 32/50
239/239 [==============================] - 24s 101ms/step - loss: 0.0526 - accuracy: 0.9852 - val_loss: 0.4056 - val_
accuracy: 0.8943
Epoch 33/50
239/239 [==============================] - 24s 101ms/step - loss: 0.0724 - accuracy: 0.9797 - val_loss: 1.6772 - val_
accuracy: 0.6981
Epoch 34/50
239/239 [==============================] - 24s 101ms/step - loss: 0.2260 - accuracy: 0.9280 - val_loss: 1.5161 - val_
accuracy: 0.7283
Epoch 35/50
239/239 [==============================] - 24s 100ms/step - loss: 0.1051 - accuracy: 0.9658 - val_loss: 0.6309 - val_
accuracy: 0.8604
Epoch 36/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1023 - accuracy: 0.9679 - val_loss: 0.9043 - val_a
ccuracy: 0.7962
Epoch 37/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0802 - accuracy: 0.9761 - val_loss: 0.7293 - val_a
ccuracy: 0.8302
Epoch 38/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0431 - accuracy: 0.9868 - val_loss: 0.5284 - val_a
ccuracy: 0.8679
Epoch 39/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0973 - accuracy: 0.9715 - val_loss: 1.5026 - val_a
ccuracy: 0.7472
Epoch 40/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1516 - accuracy: 0.9517 - val_loss: 0.9917 - val_a
ccuracy: 0.7849
Epoch 41/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1497 - accuracy: 0.9511 - val_loss: 1.0229 - val_a
ccuracy: 0.8226
Epoch 42/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0913 - accuracy: 0.9719 - val_loss: 0.5957 - val_a
ccuracy: 0.8830
```

```
Epoch 43/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0611 - accuracy: 0.9816 - val_loss: 0.6509 - val_a
ccuracy: 0.8755
Epoch 44/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0448 - accuracy: 0.9865 - val_loss: 0.5479 - val_a
ccuracy: 0.8906
Epoch 45/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0280 - accuracy: 0.9917 - val_loss: 0.1835 - val_a
ccuracy: 0.9585
Epoch 46/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0165 - accuracy: 0.9972 - val_loss: 0.3218 - val_a
ccuracy: 0.9321
Epoch 47/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0527 - accuracy: 0.9849 - val_loss: 1.0849 - val_a
ccuracy: 0.7887
Epoch 48/50
239/239 [==============================] - 23s 97ms/step - loss: 0.3158 - accuracy: 0.9045 - val_loss: 2.1013 - val_a
ccuracy: 0.6868
Epoch 49/50
239/239 [==============================] - 23s 97ms/step - loss: 0.1153 - accuracy: 0.9641 - val_loss: 0.5746 - val_a
ccuracy: 0.9057
Epoch 50/50
239/239 [==============================] - 23s 97ms/step - loss: 0.0207 - accuracy: 0.9940 - val_loss: 0.2597 - val_a
ccuracy: 0.9396
Best epoch: 45
```

In [54]:

```python
# Re-instantiate the hypermodel and train it with the optimal number of epochs from above.
hypermodel_rs = tuner_rs.hypermodel.build(best_hps_hyp_rs)
hypermodel_rs.compile(optimizer=keras.optimizers.Adam(learning_rate=best_hps_hyp_rs.get('learning_rate')),
                loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                metrics=['accuracy'])
# Retrain the model
final_hist_hyp_rs = hypermodel_rs.fit(train_ds, epochs=best_epoch, validation_data = val_ds)
```

```
Epoch 1/45
239/239 [==============================] - 35s 107ms/step - loss: 4.6554 - accuracy: 0.0470 - val_loss: 19238.4219 -
val_accuracy: 0.0000e+00
Epoch 2/45
239/239 [==============================] - 24s 100ms/step - loss: 3.6611 - accuracy: 0.1327 - val_loss: 49.3472 - val
_accuracy: 0.0340
Epoch 3/45
239/239 [==============================] - 24s 100ms/step - loss: 3.1834 - accuracy: 0.2219 - val_loss: 9.7791 - val_
accuracy: 0.0566
Epoch 4/45
239/239 [==============================] - 24s 101ms/step - loss: 2.8172 - accuracy: 0.2971 - val_loss: 3.5670 - val_
accuracy: 0.2453
Epoch 5/45
239/239 [==============================] - 23s 98ms/step - loss: 2.5289 - accuracy: 0.3986 - val_loss: 4.1538 - val_a
ccuracy: 0.2453
Epoch 6/45
239/239 [==============================] - 23s 96ms/step - loss: 2.2205 - accuracy: 0.4873 - val_loss: 6.3383 - val_a
ccuracy: 0.1811
Epoch 7/45
239/239 [==============================] - 23s 96ms/step - loss: 1.9008 - accuracy: 0.5584 - val_loss: 4.0487 - val_a
ccuracy: 0.3057
Epoch 8/45
239/239 [==============================] - 23s 96ms/step - loss: 1.6842 - accuracy: 0.6013 - val_loss: 6.7021 - val_a
ccuracy: 0.2226
Epoch 9/45
239/239 [==============================] - 23s 96ms/step - loss: 1.4968 - accuracy: 0.6443 - val_loss: 2.8965 - val_a
ccuracy: 0.4717
Epoch 10/45
239/239 [==============================] - 23s 96ms/step - loss: 1.3519 - accuracy: 0.6772 - val_loss: 2.3589 - val_a
ccuracy: 0.4830
Epoch 11/45
239/239 [==============================] - 23s 96ms/step - loss: 1.2334 - accuracy: 0.7100 - val_loss: 1.9570 - val_a
ccuracy: 0.5811
Epoch 12/45
239/239 [==============================] - 23s 96ms/step - loss: 1.1228 - accuracy: 0.7394 - val_loss: 3.1450 - val_a
ccuracy: 0.4717
Epoch 13/45
239/239 [==============================] - 23s 96ms/step - loss: 1.0419 - accuracy: 0.7593 - val_loss: 2.4890 - val_a
ccuracy: 0.5811
Epoch 14/45
239/239 [==============================] - 23s 96ms/step - loss: 0.9294 - accuracy: 0.7913 - val_loss: 1.3747 - val_a
ccuracy: 0.7245
```

```
Epoch 15/45
239/239 [==============================] - 23s 96ms/step - loss: 0.8603 - accuracy: 0.8063 - val_loss: 1.3720 - val_a
ccuracy: 0.7396
Epoch 16/45
239/239 [==============================] - 23s 96ms/step - loss: 0.7998 - accuracy: 0.8237 - val_loss: 1.2791 - val_a
ccuracy: 0.7472
Epoch 17/45
239/239 [==============================] - 23s 96ms/step - loss: 0.7248 - accuracy: 0.8463 - val_loss: 1.6655 - val_a
ccuracy: 0.6377
Epoch 18/45
239/239 [==============================] - 23s 96ms/step - loss: 0.6630 - accuracy: 0.8642 - val_loss: 1.5576 - val_a
ccuracy: 0.6830
Epoch 19/45
239/239 [==============================] - 23s 96ms/step - loss: 0.6036 - accuracy: 0.8792 - val_loss: 2.7601 - val_a
ccuracy: 0.5736
Epoch 20/45
239/239 [==============================] - 23s 96ms/step - loss: 0.5937 - accuracy: 0.8821 - val_loss: 0.9456 - val_a
ccuracy: 0.8113
Epoch 21/45
239/239 [==============================] - 23s 96ms/step - loss: 0.3305 - accuracy: 0.9208 - val_loss: 2.5177 - val_a
ccuracy: 0.6038
Epoch 22/45
239/239 [==============================] - 23s 96ms/step - loss: 0.2997 - accuracy: 0.9083 - val_loss: 1.3854 - val_a
ccuracy: 0.7019
Epoch 23/45
239/239 [==============================] - 23s 96ms/step - loss: 0.2415 - accuracy: 0.9272 - val_loss: 2.1839 - val_a
ccuracy: 0.6604
Epoch 24/45
239/239 [==============================] - 24s 101ms/step - loss: 0.1916 - accuracy: 0.9410 - val_loss: 0.9870 - val_
accuracy: 0.7547
Epoch 25/45
239/239 [==============================] - 24s 101ms/step - loss: 0.1212 - accuracy: 0.9633 - val_loss: 1.7176 - val_
accuracy: 0.6792
Epoch 26/45
239/239 [==============================] - 24s 101ms/step - loss: 0.1016 - accuracy: 0.9714 - val_loss: 2.2547 - val_
accuracy: 0.7170
Epoch 27/45
239/239 [==============================] - 24s 101ms/step - loss: 0.2617 - accuracy: 0.9162 - val_loss: 1.4640 - val_
accuracy: 0.7208
Epoch 28/45
239/239 [==============================] - 24s 100ms/step - loss: 0.1906 - accuracy: 0.9384 - val_loss: 0.9704 - val_
accuracy: 0.8038
```

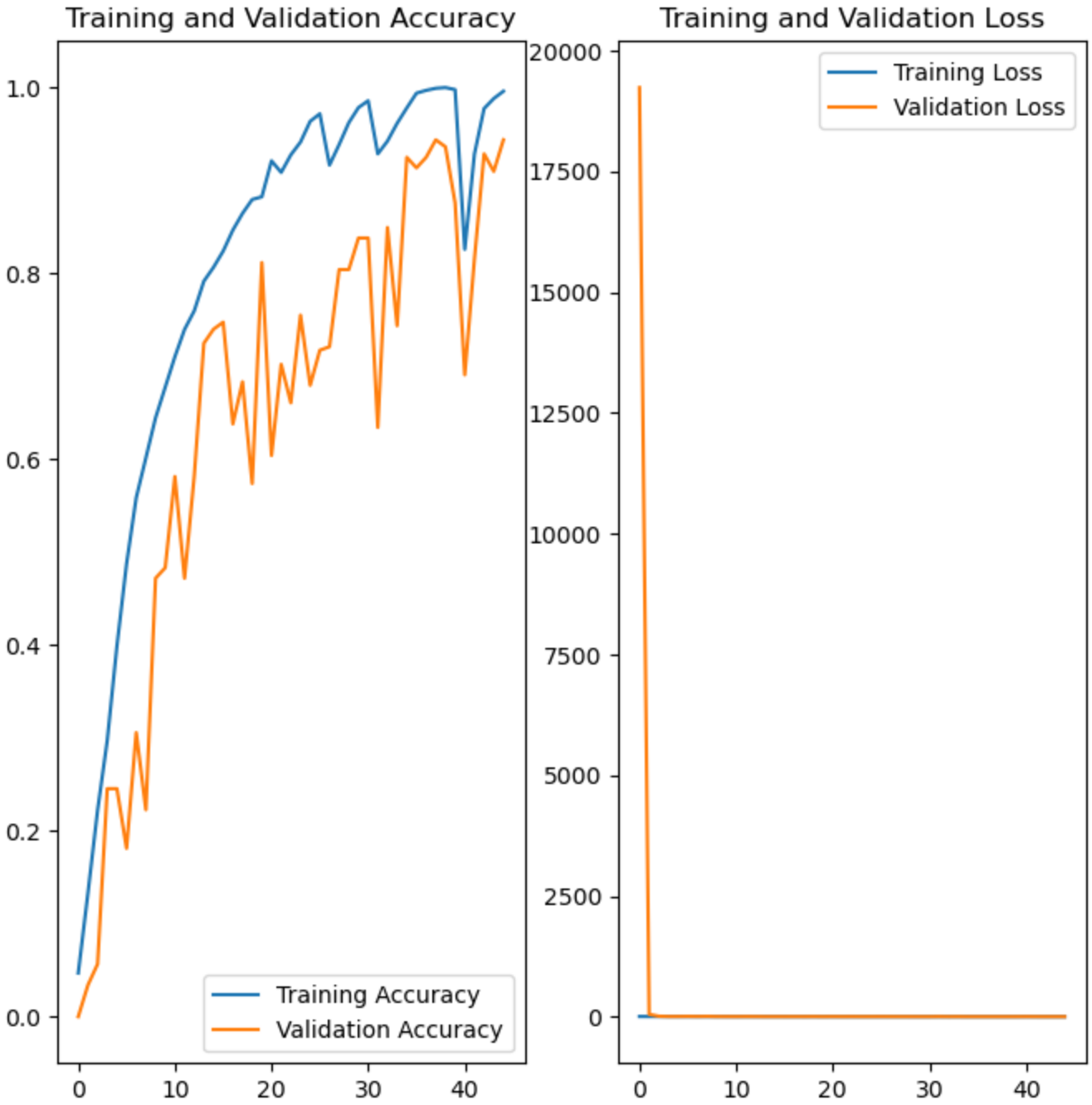
```
Epoch 29/45
239/239 [==============================] - 24s 101ms/step - loss: 0.1269 - accuracy: 0.9618 - val_loss: 1.0205 - val_
accuracy: 0.8038
Epoch 30/45
239/239 [==============================] - 24s 99ms/step - loss: 0.0788 - accuracy: 0.9781 - val_loss: 0.6584 - val_a
ccuracy: 0.8377
Epoch 31/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0539 - accuracy: 0.9854 - val_loss: 0.5579 - val_a
ccuracy: 0.8377
Epoch 32/45
239/239 [==============================] - 23s 96ms/step - loss: 0.2256 - accuracy: 0.9283 - val_loss: 2.8216 - val_a
ccuracy: 0.6340
Epoch 33/45
239/239 [==============================] - 23s 96ms/step - loss: 0.1888 - accuracy: 0.9420 - val_loss: 0.7023 - val_a
ccuracy: 0.8491
Epoch 34/45
239/239 [==============================] - 23s 96ms/step - loss: 0.1187 - accuracy: 0.9612 - val_loss: 1.6246 - val_a
ccuracy: 0.7434
Epoch 35/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0753 - accuracy: 0.9777 - val_loss: 0.3893 - val_a
ccuracy: 0.9245
Epoch 36/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0289 - accuracy: 0.9936 - val_loss: 0.3197 - val_a
ccuracy: 0.9132
Epoch 37/45
239/239 [==============================] - 23s 97ms/step - loss: 0.0151 - accuracy: 0.9966 - val_loss: 0.3831 - val_a
ccuracy: 0.9245
Epoch 38/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0058 - accuracy: 0.9988 - val_loss: 0.2582 - val_a
ccuracy: 0.9434
Epoch 39/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0065 - accuracy: 0.9996 - val_loss: 0.3088 - val_a
ccuracy: 0.9358
Epoch 40/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0120 - accuracy: 0.9974 - val_loss: 0.7758 - val_a
ccuracy: 0.8755
Epoch 41/45
239/239 [==============================] - 23s 96ms/step - loss: 0.6114 - accuracy: 0.8256 - val_loss: 2.0020 - val_a
ccuracy: 0.6906
Epoch 42/45
239/239 [==============================] - 23s 96ms/step - loss: 0.2271 - accuracy: 0.9281 - val_loss: 0.8667 - val_a
ccuracy: 0.8151
```

```
Epoch 43/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0794 - accuracy: 0.9770 - val_loss: 0.4176 - val_a
ccuracy: 0.9283
Epoch 44/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0445 - accuracy: 0.9878 - val_loss: 0.4196 - val_a
ccuracy: 0.9094
Epoch 45/45
239/239 [==============================] - 23s 96ms/step - loss: 0.0204 - accuracy: 0.9957 - val_loss: 0.2326 - val_a
ccuracy: 0.9434
```

In [59]:

```python
acc = final_hist_hyp_rs.history['accuracy']
val_acc = final_hist_hyp_rs.history['val_accuracy']
print(len(acc))
loss = final_hist_hyp_rs.history['loss']
val_loss = final_hist_hyp_rs.history['val_loss']

epochs_range = range(best_epoch)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
45
```


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss

In [57]:
```
best_val_acc = max(val_acc_per_epoch)
print(best_val_acc)
```

```
0.9584905505180359
```

In [61]:
```
print(val_loss)
```

```
[19238.421875, 49.34721374511719, 9.779139518737793, 3.5670363903045654, 4.153811454772949, 6.338341236114502, 4.0487
31327056885, 6.7021331787109375, 2.8964850902557373, 2.3589112758636475, 1.956958293914795, 3.1449942588806152, 2.488
9564514160156, 1.3746660947799683, 1.3719632625579834, 1.2791045904159546, 1.6654750108718872, 1.5575944185256958, 2.
7601399421691895, 0.945551335811615, 2.5176939964294434, 1.3853909969329834, 2.1838972568511963, 0.9869624376296997,
1.717570424079895, 2.254672050476074, 1.4640387296676636, 0.9703810214996338, 1.0205063819885254, 0.6584292650222778,
0.5578811764717102, 2.8216428756713867, 0.7023236155509949, 1.6246311664581299, 0.3892669975757599, 0.319659590721130
37, 0.38308537006378174, 0.25823670625686646, 0.30876436829566956, 0.7758459448814392, 2.0019819736480713, 0.86667662
85896301, 0.4176035523414612, 0.4196210205554962, 0.23262417316436768]
```

In [62]:
```
eval_result = hypermodel_rs.evaluate(test_ds)
print("[test loss, test accuracy]:", eval_result)
```

```
239/239 [==============================] - 13s 50ms/step - loss: 0.0275 - accuracy: 0.9919
[test loss, test accuracy]: [0.027490193024277687, 0.9918677806854248]
```