Kelompok 6

1. Imam Chalish Rafidhul Haque
2. Rakha Dhifiargo Hariadi
3. Salma Ghaida
4. Themy Sabri Syuhada

Image Classification of Hand Gestures

Pada tugas kali ini kami melakukan klasifikasi gambar dengan mengambil referensi dari permainan dua tangan orang yaitu batu-gunting-kertas atau rock-paper-scissors. Disini datanya adalah tangan-tangan asli yang membentuk gestur menyerupai batu atau gunting atau kertas. Jadi akan mengklasifikasikan sebuah gambar nantinya apakah masuk ke kategori batu, gunting, atau kertas.

Melakukan Import library-library yang akan dipakai

In [1]:
```
import tensorflow as tf
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from google.colab import files
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

Impor dataset

In [2]:
```
!wget --no-check-certificate \
  https://github.com/ThemySabri/ImageClassification/releases/download/filedata/tanganfin
  -O /content/dataTangan.zip
```

```
--2023-11-26 16:41:57--  https://github.com/ThemySabri/ImageClassification/releases/down
load/filedata/tanganfinal.zip
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/7
23629701/68231533-23eb-4e2b-acea-7d673dd7d245?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Cre
dential=AKIAIWNJYAX4CSVEH53A%2F20231126%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231
126T164157Z&X-Amz-Expires=300&X-Amz-Signature=c4609d3766812acad9511644dd36b675ad4f37f6c7
46cf5eaba4e9ed84685ceb&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=723629701&re
sponse-content-disposition=attachment%3B%20filename%3Dtanganfinal.zip&response-content-t
ype=application%2Foctet-stream [following]
--2023-11-26 16:41:57--  https://objects.githubusercontent.com/github-production-release
-asset-2e65be/723629701/68231533-23eb-4e2b-acea-7d673dd7d245?X-Amz-Algorithm=AWS4-HMAC-S
HA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20231126%2Fus-east-1%2Fs3%2Faws4_request&X
-Amz-Date=20231126T164157Z&X-Amz-Expires=300&X-Amz-Signature=c4609d3766812acad9511644dd3
6b675ad4f37f6c746cf5eaba4e9ed84685ceb&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_
id=723629701&response-content-disposition=attachment%3B%20filename%3Dtanganfinal.zip&res
ponse-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.1
```

```
33, 185.199.108.133, 185.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 254120840 (242M) [application/octet-stream]
Saving to: '/content/dataTangan.zip'

/content/dataTangan 100%[===================>] 242.35M   240MB/s    in 1.0s

2023-11-26 16:41:58 (240 MB/s) - '/content/dataTangan.zip' saved [254120840/254120840]
```

Melakukan ekstraksi zip

In [3]:
```python
import zipfile,os
local_zip = '/content/dataTangan.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content/all/')
zip_ref.close()
```

Mengecek setiap data yang sudah diekstrak

In [4]:
```python
Batu = os.listdir('/content/all/Batu')
Kertas = os.listdir('/content/all/Kertas')
Gunting = os.listdir('/content/all/Gunting')

lenall = len(Batu) + len(Gunting) + len(Kertas)
print(f"Banyaknya data gambar Batu: {len(Batu)}")
print(f"Banyaknya data gambar Kertas: {len(Gunting)}")
print(f"Banyaknya data gambar Gunting: {len(Kertas)}")
print(f"Banyaknya data gambar keseluruhan: {lenall}")
```

```
Banyaknya data gambar Batu: 54
Banyaknya data gambar Kertas: 60
Banyaknya data gambar Gunting: 60
Banyaknya data gambar keseluruhan: 174
```

Melakukan pemisahan data train dan validation dengan rasio 75% di train dan 25% di validation

In [5]:
```python
import os
import shutil

# Path ke direktori utama
main_directory = '/content'

# List kategori
categories = ['Batu', 'Gunting', 'Kertas']

# Membuat folder utama (train dan val)
train_main_directory = os.path.join(main_directory, 'train')
val_main_directory = os.path.join(main_directory, 'val')

os.makedirs(train_main_directory, exist_ok=True)
os.makedirs(val_main_directory, exist_ok=True)

# Hitung proporsi sebelum pembulatan
train_ratio = 0.75

train_size = round(lenall * train_ratio)

# Iterasi melalui setiap kategori
```

```
for category in categories:
    category_path = os.path.join(main_directory, "all/", category)

    # Path untuk data train dan data validation dalam setiap kategori
    train_category_directory = os.path.join(train_main_directory, category)
    val_category_directory = os.path.join(val_main_directory, category)

    # Membuat subfolder rock, paper, scissors di dalam folder train dan val
    os.makedirs(train_category_directory, exist_ok=True)
    os.makedirs(val_category_directory, exist_ok=True)

    # Mendapatkan list file dalam setiap kategori
    files = os.listdir(category_path)

    # Menghitung jumlah data train berdasarkan proporsi tanpa pembulatan
    train_size_category = round(len(files) * train_ratio)

    # Memindahkan file ke subfolder train
    for file in files[:train_size_category]:
        src_path = os.path.join(category_path, file)
        dest_path = os.path.join(train_category_directory, file)
        shutil.copy(src_path, dest_path)

    # Memindahkan file ke subfolder val
    for file in files[train_size_category:]:
        src_path = os.path.join(category_path, file)
        dest_path = os.path.join(val_category_directory, file)
        shutil.copy(src_path, dest_path)

train1 = os.listdir('/content/train/Batu')
train2 = os.listdir('/content/train/Kertas')
train3 = os.listdir('/content/train/Gunting')

lentrain = len(train1) + len(train2) + len(train3)

val1 = os.listdir('/content/val/Batu')
val2 = os.listdir('/content/val/Kertas')
val3 = os.listdir('/content/val/Gunting')

lenval = len(val1) + len(val2) + len(val3)

print(f"Total data gambar training: {lentrain}")
print(f"Total data gambar validation: {lenval}")
```

```
Total data gambar training: 130
Total data gambar validation: 44
```

Mengecek lagi data yang sudah dipisah

In [6]:
```
base_dir = '/content'
train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'val')
```

In [7]:
```
os.listdir(train_dir)
```

Out[7]:
```
['Kertas', 'Gunting', 'Batu']
```

In [8]:
```
os.listdir(val_dir)
```

Out[8]:
```
['Kertas', 'Gunting', 'Batu']
```

Melakukan utak-atik pada model

Yang akan diutak-atik disini adalah augmentasi gambar dan juga pembuatan sequential modelnya terutama di layeringnya.

Model pertama

Pada model ini convolution serta pooling akan dilakukan sebanyak 4 kali dengan parameter yang bisa dilihat dibawah, lalu juga model ini tidak menjalankan augmentasi gambar sama sekali, hanya melakukan rescaling.

In [9]:
```
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

In [10]:
```
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 256)       295168

 max_pooling2d_3 (MaxPoolin  (None, 7, 7, 256)         0
 g2D)

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 512)               6423040

 dense_1 (Dense)             (None, 3)                 1539

=================================================================
Total params: 6812995 (25.99 MB)
Trainable params: 6812995 (25.99 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [11]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
```

```
model1.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'
```

Melakukan augmentasi gambar pada data train, validasi tidak perlu

In [12]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)

validation_datagen = ImageDataGenerator(rescale=1./255)
```

Menyiapkan data latih dan validasi dari kumpulan data gambar tadi untuk dipakai pelatihan model

In [13]:
```
train_generator = train_datagen.flow_from_directory(
        train_dir,  # direktori data latih
        target_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piks
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        val_dir, # direktori data validasi
        target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 pikse
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')
```

```
Found 130 images belonging to 3 classes.
Found 44 images belonging to 3 classes.
```

Melatih model

In [14]:
```
history1 = model1.fit_generator(
    train_generator,
    steps_per_epoch=16,
    epochs=15,
    validation_data=validation_generator,
    validation_steps=4
)
```

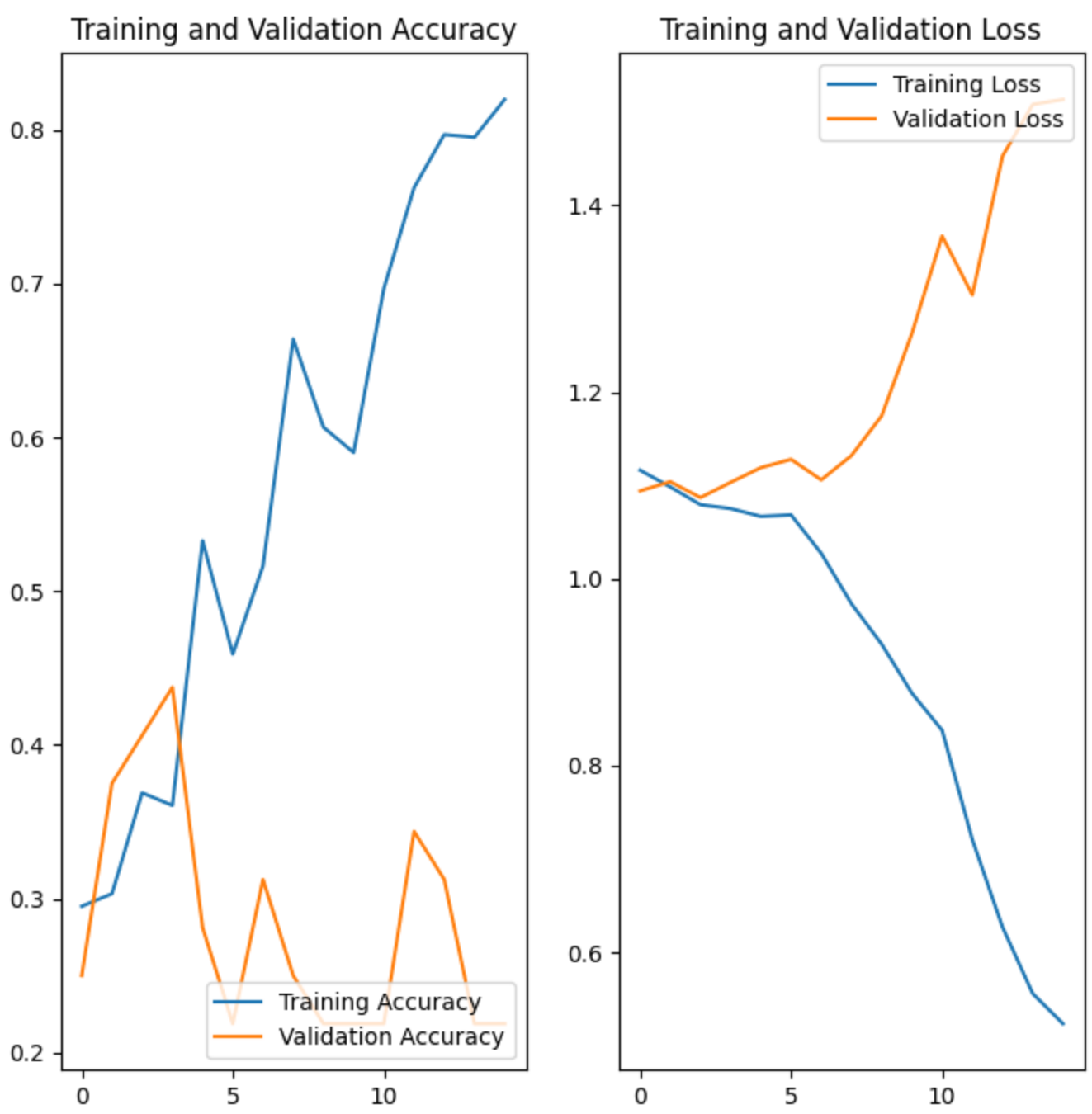
```
Epoch 1/15
16/16 [==============================] - 22s 1s/step - loss: 1.1164 - accuracy: 0.2951 -
val_loss: 1.0942 - val_accuracy: 0.2500
Epoch 2/15
16/16 [==============================] - 20s 1s/step - loss: 1.0985 - accuracy: 0.3033 -
val_loss: 1.1041 - val_accuracy: 0.3750
Epoch 3/15
16/16 [==============================] - 18s 1s/step - loss: 1.0794 - accuracy: 0.3689 -
val_loss: 1.0871 - val_accuracy: 0.4062
Epoch 4/15
16/16 [==============================] - 18s 1s/step - loss: 1.0752 - accuracy: 0.3607 -
val_loss: 1.1034 - val_accuracy: 0.4375
Epoch 5/15
16/16 [==============================] - 25s 2s/step - loss: 1.0669 - accuracy: 0.5328 -
val_loss: 1.1191 - val_accuracy: 0.2812
Epoch 6/15
16/16 [==============================] - 18s 1s/step - loss: 1.0685 - accuracy: 0.4590 -
val_loss: 1.1280 - val_accuracy: 0.2188
Epoch 7/15
16/16 [==============================] - 20s 1s/step - loss: 1.0273 - accuracy: 0.5164 -
val_loss: 1.1060 - val_accuracy: 0.3125
Epoch 8/15
16/16 [==============================] - 19s 1s/step - loss: 0.9732 - accuracy: 0.6639 -
```

```
val_loss: 1.1321 - val_accuracy: 0.2500
Epoch 9/15
16/16 [==============================] - 19s 1s/step - loss: 0.9299 - accuracy: 0.6066 -
val_loss: 1.1747 - val_accuracy: 0.2188
Epoch 10/15
16/16 [==============================] - 19s 1s/step - loss: 0.8775 - accuracy: 0.5902 -
val_loss: 1.2629 - val_accuracy: 0.2188
Epoch 11/15
16/16 [==============================] - 19s 1s/step - loss: 0.8378 - accuracy: 0.6967 -
val_loss: 1.3672 - val_accuracy: 0.2188
Epoch 12/15
16/16 [==============================] - 20s 1s/step - loss: 0.7206 - accuracy: 0.7623 -
val_loss: 1.3042 - val_accuracy: 0.3438
Epoch 13/15
16/16 [==============================] - 18s 1s/step - loss: 0.6268 - accuracy: 0.7969 -
val_loss: 1.4530 - val_accuracy: 0.3125
Epoch 14/15
16/16 [==============================] - 19s 1s/step - loss: 0.5556 - accuracy: 0.7951 -
val_loss: 1.5082 - val_accuracy: 0.2188
Epoch 15/15
16/16 [==============================] - 19s 1s/step - loss: 0.5234 - accuracy: 0.8197 -
val_loss: 1.5135 - val_accuracy: 0.2188
```

Plot hasil training

In [15]:

```python
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']
loss = history1.history['loss']
val_loss = history1.history['val_loss']
epochs_range = range(15)
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()
```



Model kedua

Pada model ini kita mengurangi 1 layer serta mengubah sedikit parameter, lalu juga model ini menjalankan augmentasi gambar yang cukup banyak.

In [16]:
```
model2 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

In [17]:
```
model2.summary()
```

```
Model: "sequential_1"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_4 (MaxPoolin  (None, 74, 74, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 conv2d_6 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_6 (MaxPoolin  (None, 17, 17, 128)       0
 g2D)

 flatten_1 (Flatten)         (None, 36992)             0

 dense_2 (Dense)             (None, 256)               9470208

 dense_3 (Dense)             (None, 3)                 771

=================================================================
Total params: 9564227 (36.48 MB)
Trainable params: 9564227 (36.48 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [18]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model2.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'
```

Melakukan augmentasi gambar pada data train, validasi tidak perlu

In [19]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
    )

validation_datagen = ImageDataGenerator(rescale=1./255)
```

Menyiapkan data latih dan validasi dari kumpulan data gambar tadi untuk dipakai pelatihan model

In [20]:
```
train_generator = train_datagen.flow_from_directory(
        train_dir,  # direktori data latih
        target_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piks
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        val_dir, # direktori data validasi
        target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 pikse
```

```
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')
```

```
Found 130 images belonging to 3 classes.
Found 44 images belonging to 3 classes.
```

Melatih model

In [21]:

```
history2 = model2.fit_generator(
    train_generator,
    steps_per_epoch=16,
    epochs=15,
    validation_data=validation_generator,
    validation_steps=4
)
```

```
Epoch 1/15
16/16 [==============================] - 20s 1s/step - loss: 1.1388 - accuracy: 0.3361 -
val_loss: 1.1155 - val_accuracy: 0.2812
Epoch 2/15
16/16 [==============================] - 19s 1s/step - loss: 1.1087 - accuracy: 0.2951 -
val_loss: 1.0973 - val_accuracy: 0.3750
Epoch 3/15
16/16 [==============================] - 17s 1s/step - loss: 1.1044 - accuracy: 0.3525 -
val_loss: 1.1073 - val_accuracy: 0.2812
Epoch 4/15
16/16 [==============================] - 18s 1s/step - loss: 1.0925 - accuracy: 0.4062 -
val_loss: 1.0910 - val_accuracy: 0.3750
Epoch 5/15
16/16 [==============================] - 20s 1s/step - loss: 1.0850 - accuracy: 0.4344 -
val_loss: 1.1203 - val_accuracy: 0.2188
Epoch 6/15
16/16 [==============================] - 19s 1s/step - loss: 1.0870 - accuracy: 0.3770 -
val_loss: 1.1196 - val_accuracy: 0.2812
Epoch 7/15
16/16 [==============================] - 18s 1s/step - loss: 1.0846 - accuracy: 0.3934 -
val_loss: 1.1658 - val_accuracy: 0.2812
Epoch 8/15
16/16 [==============================] - 19s 1s/step - loss: 1.1033 - accuracy: 0.3770 -
val_loss: 1.1185 - val_accuracy: 0.3125
Epoch 9/15
16/16 [==============================] - 20s 1s/step - loss: 1.0814 - accuracy: 0.3770 -
val_loss: 1.1114 - val_accuracy: 0.4062
Epoch 10/15
16/16 [==============================] - 18s 1s/step - loss: 1.0796 - accuracy: 0.4262 -
val_loss: 1.1247 - val_accuracy: 0.3125
Epoch 11/15
16/16 [==============================] - 17s 1s/step - loss: 1.0813 - accuracy: 0.4016 -
val_loss: 1.1468 - val_accuracy: 0.3125
Epoch 12/15
16/16 [==============================] - 18s 1s/step - loss: 1.0534 - accuracy: 0.4672 -
val_loss: 1.1043 - val_accuracy: 0.4688
Epoch 13/15
16/16 [==============================] - 17s 1s/step - loss: 1.0807 - accuracy: 0.3852 -
val_loss: 1.1134 - val_accuracy: 0.3438
Epoch 14/15
16/16 [==============================] - 18s 1s/step - loss: 1.0571 - accuracy: 0.4344 -
val_loss: 1.1433 - val_accuracy: 0.3438
Epoch 15/15
16/16 [==============================] - 20s 1s/step - loss: 1.0614 - accuracy: 0.3934 -
val_loss: 1.0813 - val_accuracy: 0.4688
```

Plot hasil training

In [22]:
```
acc = history2.history['accuracy']
val_acc = history2.history['val_accuracy']
loss = history2.history['loss']
val_loss = history2.history['val_loss']
epochs_range = range(15)
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()
```

Model ketiga

Pada model ini kita menggunakan metode transfer learning yang dimana menggunakan pre-trained model yaitu VGG16, lalu disini juga menambah droput. Di model ini tidak melakukan augmentasi gambar sama sekali, hanya rescaling.

In [23]:
```
from tensorflow.keras.applications import VGG16
# Load pre-trained VGG16 model (tanpa lapisan dense teratas)
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
# Freeze the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

model3 = tf.keras.models.Sequential([
    base_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
6/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

In [24]:
```
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten_2 (Flatten)         (None, 8192)              0

 dense_4 (Dense)             (None, 512)               4194816

 dropout (Dropout)           (None, 512)               0

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 dense_5 (Dense)             (None, 3)                 1539

=================================================================
Total params: 18913091 (72.15 MB)
Trainable params: 4197379 (16.01 MB)
Non-trainable params: 14715712 (56.14 MB)
_________________________________________________________________
```

In [25]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model3.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'
```

Melakukan augmentasi gambar pada data train, validasi tidak perlu

In [26]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)

validation_datagen = ImageDataGenerator(rescale=1./255)
```

Menyiapkan data latih dan validasi dari kumpulan data gambar tadi untuk dipakai pelatihan model

In [27]:
```
train_generator = train_datagen.flow_from_directory(
        train_dir,  # direktori data latih
        target_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piks
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        val_dir, # direktori data validasi
        target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 pikse
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')
```

```
Found 130 images belonging to 3 classes.
Found 44 images belonging to 3 classes.
```

Melatih model

In [28]:
```
history3 = model3.fit_generator(
    train_generator,
    steps_per_epoch=16,
    epochs=15,
    validation_data=validation_generator,
    validation_steps=4
)
```

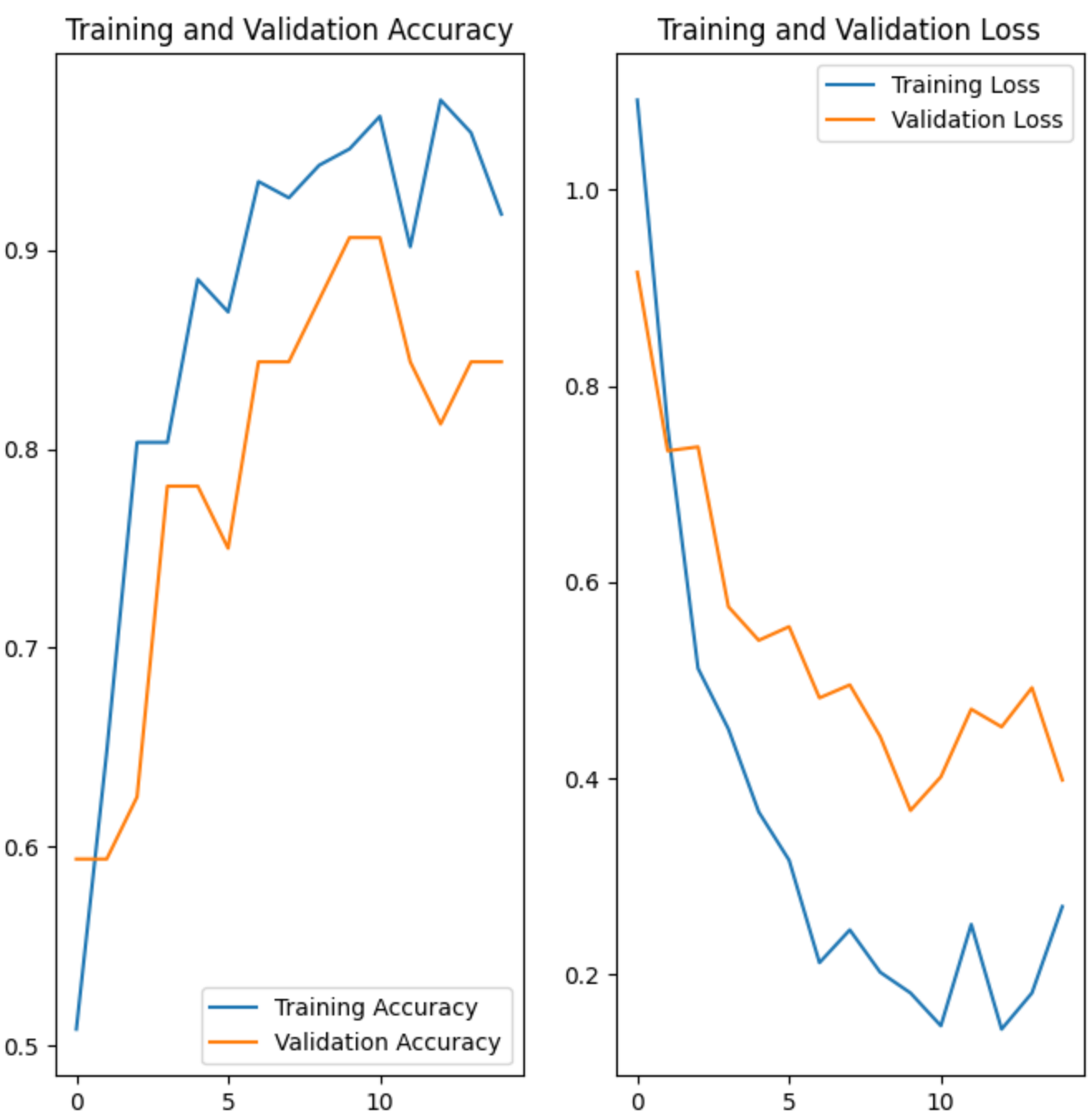
```
Epoch 1/15
16/16 [==============================] - 53s 3s/step - loss: 1.0917 - accuracy: 0.5082 -
val_loss: 0.9161 - val_accuracy: 0.5938
Epoch 2/15
16/16 [==============================] - 62s 4s/step - loss: 0.7582 - accuracy: 0.6475 -
val_loss: 0.7339 - val_accuracy: 0.5938
Epoch 3/15
16/16 [==============================] - 60s 4s/step - loss: 0.5120 - accuracy: 0.8033 -
val_loss: 0.7380 - val_accuracy: 0.6250
Epoch 4/15
16/16 [==============================] - 59s 4s/step - loss: 0.4502 - accuracy: 0.8033 -
val_loss: 0.5750 - val_accuracy: 0.7812
Epoch 5/15
16/16 [==============================] - 59s 4s/step - loss: 0.3656 - accuracy: 0.8852 -
val_loss: 0.5407 - val_accuracy: 0.7812
Epoch 6/15
16/16 [==============================] - 60s 4s/step - loss: 0.3164 - accuracy: 0.8689 -
val_loss: 0.5546 - val_accuracy: 0.7500
Epoch 7/15
16/16 [==============================] - 51s 3s/step - loss: 0.2119 - accuracy: 0.9344 -
val_loss: 0.4819 - val_accuracy: 0.8438
Epoch 8/15
16/16 [==============================] - 61s 4s/step - loss: 0.2454 - accuracy: 0.9262 -
val_loss: 0.4952 - val_accuracy: 0.8438
Epoch 9/15
16/16 [==============================] - 61s 4s/step - loss: 0.2022 - accuracy: 0.9426 -
val_loss: 0.4426 - val_accuracy: 0.8750
Epoch 10/15
16/16 [==============================] - 61s 4s/step - loss: 0.1810 - accuracy: 0.9508 -
val_loss: 0.3672 - val_accuracy: 0.9062
Epoch 11/15
16/16 [==============================] - 60s 4s/step - loss: 0.1475 - accuracy: 0.9672 -
val_loss: 0.4015 - val_accuracy: 0.9062
Epoch 12/15
16/16 [==============================] - 59s 4s/step - loss: 0.2510 - accuracy: 0.9016 -
val_loss: 0.4704 - val_accuracy: 0.8438
Epoch 13/15
16/16 [==============================] - 59s 4s/step - loss: 0.1441 - accuracy: 0.9754 -
```

```
val_loss: 0.4524 - val_accuracy: 0.8125
Epoch 14/15
16/16 [==============================] - 63s 4s/step - loss: 0.1810 - accuracy: 0.9590 -
val_loss: 0.4923 - val_accuracy: 0.8438
Epoch 15/15
16/16 [==============================] - 59s 4s/step - loss: 0.2691 - accuracy: 0.9180 -
val_loss: 0.3982 - val_accuracy: 0.8438
```

Plot hasil training

In [29]:

```
acc = history3.history['accuracy']
val_acc = history3.history['val_accuracy']
loss = history3.history['loss']
val_loss = history3.history['val_loss']
epochs_range = range(15)
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()
```



Model keempat

Pada model ini sama seperti sebelumnnya, kita menggunakan metode transfer learning yang dimana menggunakan pre-trained model yaitu VGG16, lalu disini juga masih ada droput. Di model ini melakukan augmentasi gambar yang sama dengan model kedua.

In [30]:
```
from tensorflow.keras.applications import VGG16
# Load pre-trained VGG16 model (tanpa lapisan dense teratas)
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
# Freeze the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

model4 = tf.keras.models.Sequential([
    base_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.BatchNormalization(),
```

```
    tf.keras.layers.Dense(3, activation='softmax')
])
```

In [31]:
```
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten_3 (Flatten)         (None, 8192)              0

 dense_6 (Dense)             (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 batch_normalization_1 (Bat  (None, 512)               2048
 chNormalization)

 dense_7 (Dense)             (None, 3)                 1539

=================================================================
Total params: 18913091 (72.15 MB)
Trainable params: 4197379 (16.01 MB)
Non-trainable params: 14715712 (56.14 MB)
_________________________________________________________________
```

In [32]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model4.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'
```

Melakukan augmentasi gambar pada data train, validasi tidak perlu

In [33]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
    )

validation_datagen = ImageDataGenerator(rescale=1./255)
```

Menyiapkan data latih dan validasi dari kumpulan data gambar tadi untuk dipakai pelatihan model

In [34]:
```
train_generator = train_datagen.flow_from_directory(
        train_dir,  # direktori data latih
        target_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piks
        batch_size=8,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        val_dir, # direktori data validasi
        target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 pikse
        batch_size=8,
```

```
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode =
        class_mode='categorical')
```

```
Found 130 images belonging to 3 classes.
Found 44 images belonging to 3 classes.
```

Melatih model

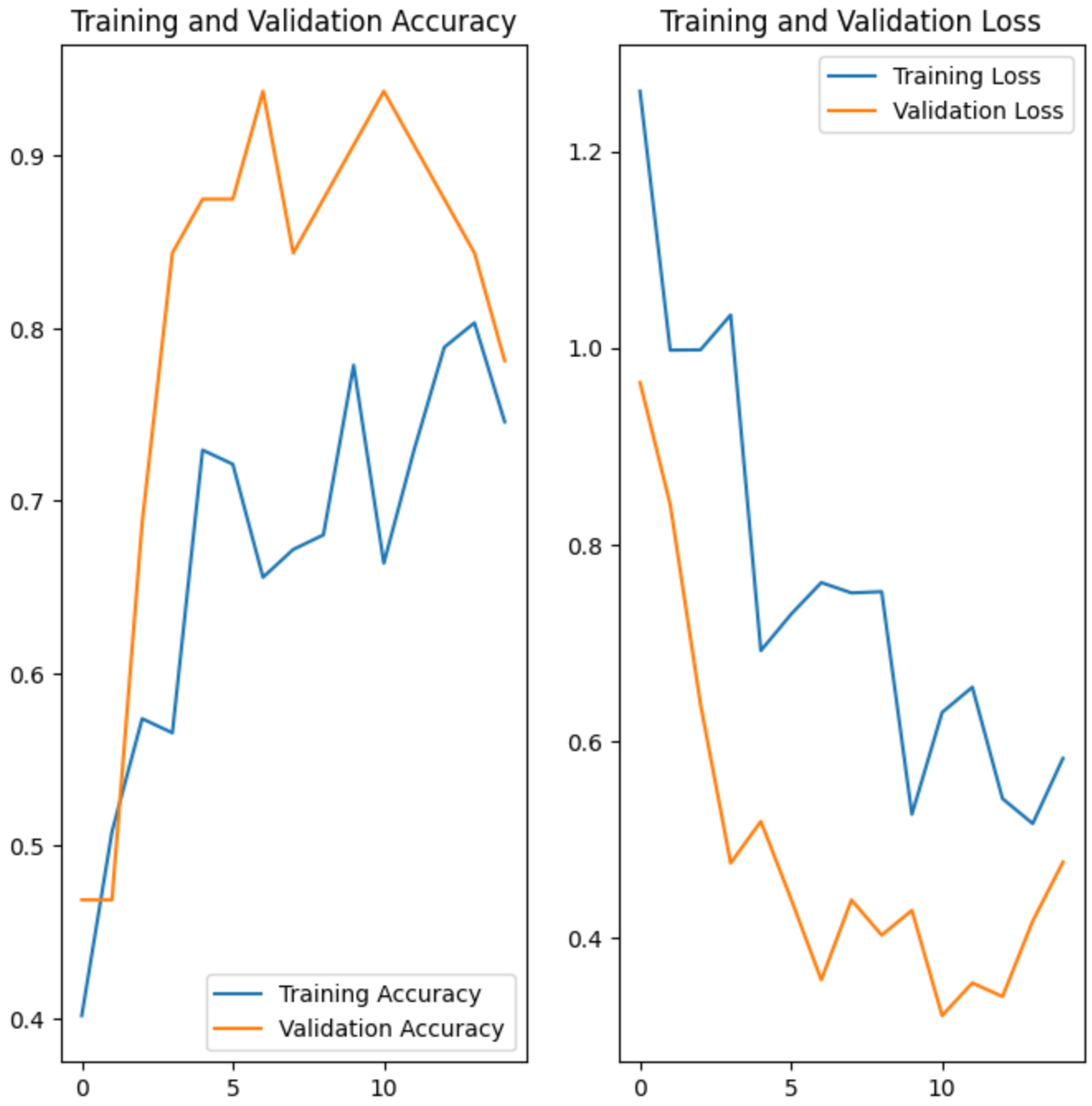
In [35]:

```
history4 = model4.fit_generator(
    train_generator,
    steps_per_epoch=16,
    epochs=15,
    validation_data=validation_generator,
    validation_steps=4
)
```

```
Epoch 1/15
16/16 [==============================] - 54s 3s/step - loss: 1.2612 - accuracy: 0.4016 -
val_loss: 0.9650 - val_accuracy: 0.4688
Epoch 2/15
16/16 [==============================] - 59s 4s/step - loss: 0.9979 - accuracy: 0.5082 -
val_loss: 0.8407 - val_accuracy: 0.4688
Epoch 3/15
16/16 [==============================] - 62s 4s/step - loss: 0.9981 - accuracy: 0.5738 -
val_loss: 0.6381 - val_accuracy: 0.6875
Epoch 4/15
16/16 [==============================] - 52s 3s/step - loss: 1.0336 - accuracy: 0.5656 -
val_loss: 0.4762 - val_accuracy: 0.8438
Epoch 5/15
16/16 [==============================] - 51s 3s/step - loss: 0.6922 - accuracy: 0.7295 -
val_loss: 0.5186 - val_accuracy: 0.8750
Epoch 6/15
16/16 [==============================] - 62s 4s/step - loss: 0.7292 - accuracy: 0.7213 -
val_loss: 0.4397 - val_accuracy: 0.8750
Epoch 7/15
16/16 [==============================] - 60s 4s/step - loss: 0.7615 - accuracy: 0.6557 -
val_loss: 0.3574 - val_accuracy: 0.9375
Epoch 8/15
16/16 [==============================] - 52s 3s/step - loss: 0.7510 - accuracy: 0.6719 -
val_loss: 0.4388 - val_accuracy: 0.8438
Epoch 9/15
16/16 [==============================] - 61s 4s/step - loss: 0.7522 - accuracy: 0.6803 -
val_loss: 0.4029 - val_accuracy: 0.8750
Epoch 10/15
16/16 [==============================] - 64s 4s/step - loss: 0.5258 - accuracy: 0.7787 -
val_loss: 0.4281 - val_accuracy: 0.9062
Epoch 11/15
16/16 [==============================] - 59s 4s/step - loss: 0.6295 - accuracy: 0.6639 -
val_loss: 0.3211 - val_accuracy: 0.9375
Epoch 12/15
16/16 [==============================] - 59s 4s/step - loss: 0.6551 - accuracy: 0.7295 -
val_loss: 0.3543 - val_accuracy: 0.9062
Epoch 13/15
16/16 [==============================] - 65s 4s/step - loss: 0.5417 - accuracy: 0.7891 -
val_loss: 0.3405 - val_accuracy: 0.8750
Epoch 14/15
16/16 [==============================] - 63s 4s/step - loss: 0.5164 - accuracy: 0.8033 -
val_loss: 0.4173 - val_accuracy: 0.8438
Epoch 15/15
16/16 [==============================] - 51s 3s/step - loss: 0.5827 - accuracy: 0.7459 -
val_loss: 0.4772 - val_accuracy: 0.7812
```

Plot hasil training

In [36]:
```
acc = history4.history['accuracy']
val_acc = history4.history['val_accuracy']
loss = history4.history['loss']
val_loss = history4.history['val_loss']
epochs_range = range(15)
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()
```



Kesimpulan model

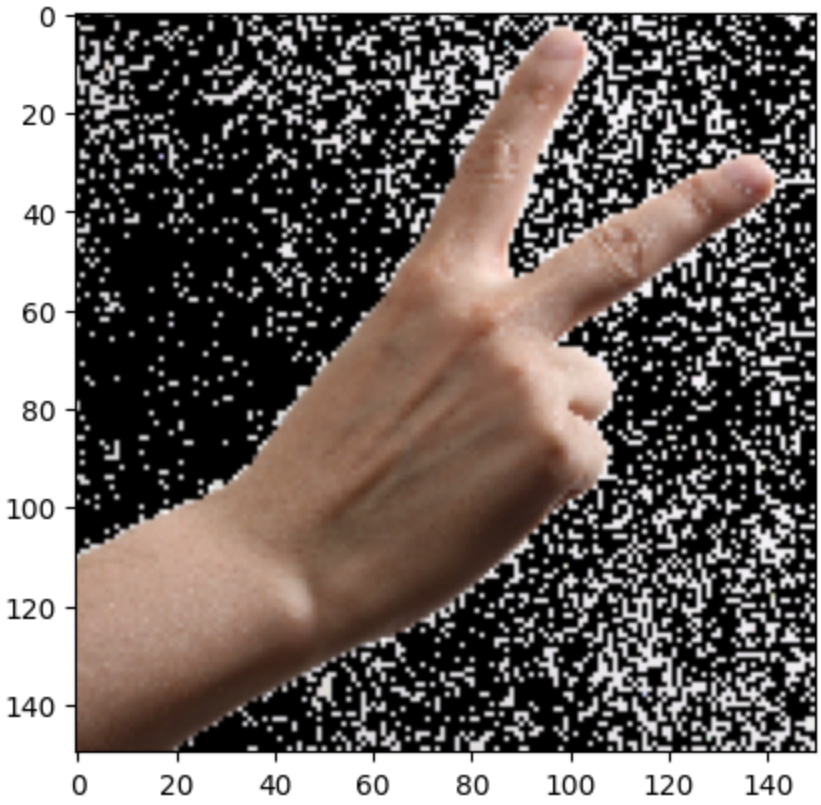
1. Model pertama sangat baik dalam training accuracy dan training loss nya, akan tetapi sangat buruk di validation accuracy dan validation loss.
2. Model kedua sangat buruk dalam segala aspek, bisa dilihat di plot nya dimana terlihat sangat kacau.
3. Model ketiga ini sangat baik dalam segala aspek, plot juga menunjukkan seberapa baik model tersebut. Di model ini training accuracy dan training loss nya lebih baik daripada validation accuracy dan validation loss nya.
4. Model keempat ini juga sangat baik dalam segala aspek, plot juga menunjukkan hal yang sama. Tetapi ada perbedaan dengan model ketiga dimana yang lebih baiknya adalah validation accuracy dan validation loss nya, bisa dibilang model ketiga dan keempat ini cukup berlawanan.

Artinya model yang harus dicari adalah jalan tengah dari model ketiga dan keempat, yang dimana semua aspek bagus serta perbedaan training dan validationnya tidak terpampang jauh.

In [39]:
```
img_width = 150
img_height = 150
image_baru_url = "https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Rock-paper-s
image_baru_path = tf.keras.utils.get_file('', origin=image_baru_url)
img = tf.keras.utils.load_img(
    image_baru_path, target_size=(img_height, img_width)
)
plt.imshow(img)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch


prediction1 = model1.predict(img_array)
prediction2 = model2.predict(img_array)
prediction3 = model3.predict(img_array)
prediction4 = model4.predict(img_array)
score1 = tf.nn.softmax(prediction1[0])
score2 = tf.nn.softmax(prediction2[0])
score3 = tf.nn.softmax(prediction3[0])
score4 = tf.nn.softmax(prediction4[0])

print("Model pertama:")
if np.argmax(score1) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score1) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model kedua:")
if np.argmax(score2) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score2) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model ketiga:")
if np.argmax(score3) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score3) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model keempat:")
if np.argmax(score4) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
```

```
elif np.argmax(score4) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo
```

```
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 115ms/step
1/1 [==============================] - 1s 511ms/step
1/1 [==============================] - 0s 444ms/step
Model pertama:
This image most likely belongs to Batu with a 57.61 percent confidence.
Model kedua:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model ketiga:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model keempat:
This image most likely belongs to Kertas with a 57.61 percent confidence.
```

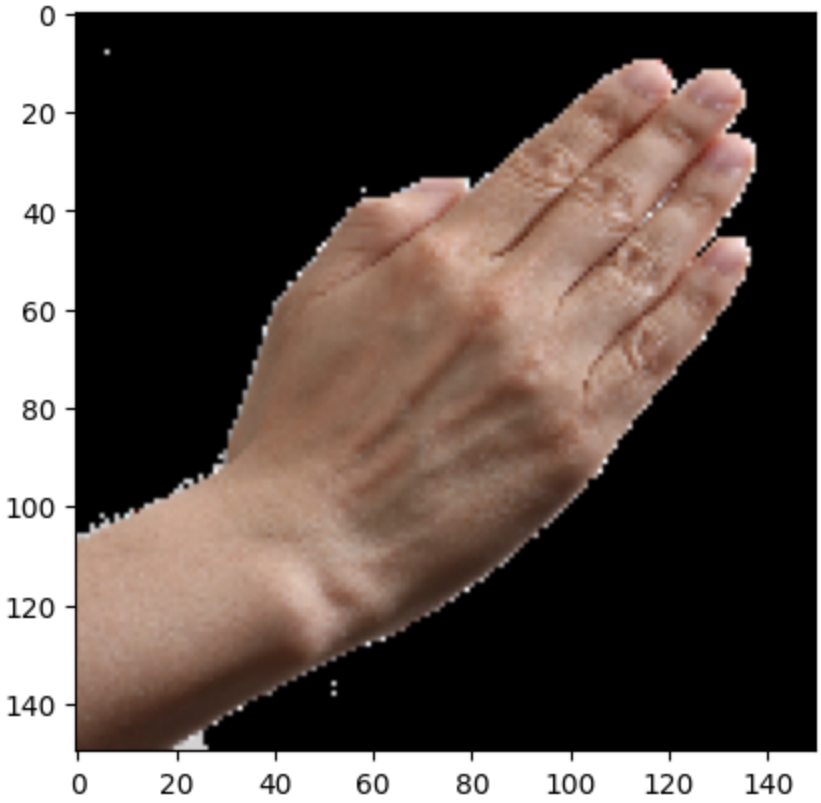


In [40]:

```
img_width = 150
img_height = 150
image_baru_url = "https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Rock-paper-s
image_baru_path = tf.keras.utils.get_file('', origin=image_baru_url)
img = tf.keras.utils.load_img(
    image_baru_path, target_size=(img_height, img_width)
)
plt.imshow(img)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch


prediction1 = model1.predict(img_array)
prediction2 = model2.predict(img_array)
prediction3 = model3.predict(img_array)
prediction4 = model4.predict(img_array)
score1 = tf.nn.softmax(prediction1[0])
score2 = tf.nn.softmax(prediction2[0])
score3 = tf.nn.softmax(prediction3[0])
score4 = tf.nn.softmax(prediction4[0])
```

```
print("Model pertama:")
if np.argmax(score1) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score1) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model kedua:")
if np.argmax(score2) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score2) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model ketiga:")
if np.argmax(score3) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score3) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model keempat:")
if np.argmax(score4) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score4) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo
```

```
Downloading data from https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Rock-pap
er-scissors_%28paper%29.png/800px-Rock-paper-scissors_%28paper%29.png
486581/486581 [==============================] - 0s 0us/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 247ms/step
1/1 [==============================] - 0s 251ms/step
Model pertama:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model kedua:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model ketiga:
This image most likely belongs to Batu with a 55.85 percent confidence.
Model keempat:
This image most likely belongs to Batu with a 57.61 percent confidence.
```

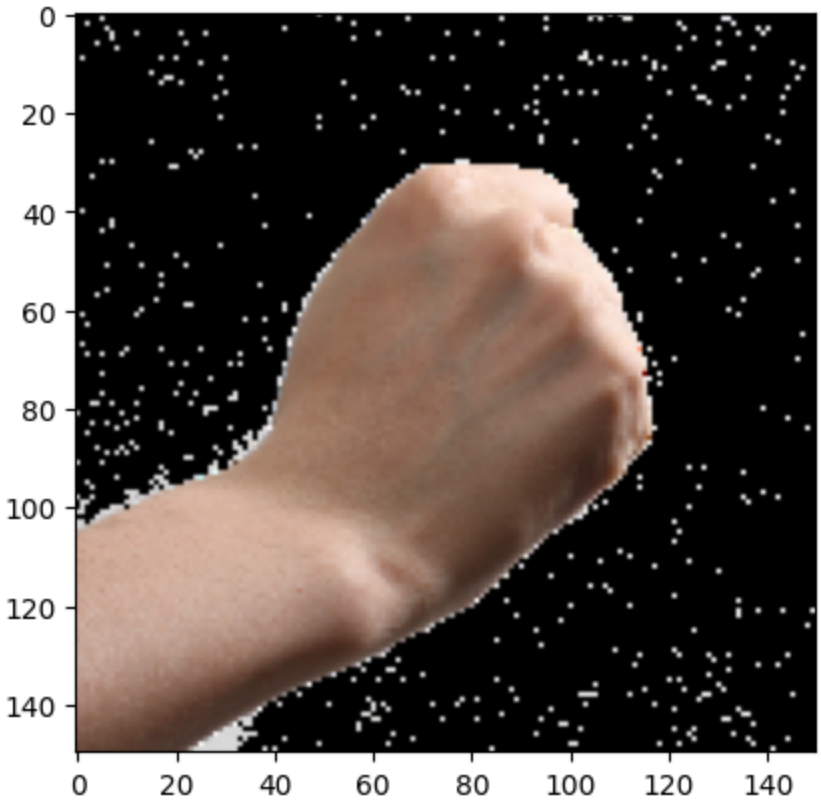


In [41]:

```
img_width = 150
img_height = 150
image_baru_url = "https://upload.wikimedia.org/wikipedia/commons/thumb/7/7e/Rock-paper-s
image_baru_path = tf.keras.utils.get_file('', origin=image_baru_url)
img = tf.keras.utils.load_img(
    image_baru_path, target_size=(img_height, img_width)
)
plt.imshow(img)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch


prediction1 = model1.predict(img_array)
prediction2 = model2.predict(img_array)
prediction3 = model3.predict(img_array)
prediction4 = model4.predict(img_array)
score1 = tf.nn.softmax(prediction1[0])
score2 = tf.nn.softmax(prediction2[0])
score3 = tf.nn.softmax(prediction3[0])
score4 = tf.nn.softmax(prediction4[0])

print("Model pertama:")
if np.argmax(score1) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score1) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model kedua:")
if np.argmax(score2) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score2) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model ketiga:")
```

```
if np.argmax(score3) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score3) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo

print("Model keempat:")
if np.argmax(score4) == 0:
  print("This image most likely belongs to Batu with a {:.2f} percent confidence.".forma
elif np.argmax(score4) == 1:
  print("This image most likely belongs to Kertas with a {:.2f} percent confidence.".for
else:
  print("This image most likely belongs to Gunting with a {:.2f} percent confidence.".fo
```

```
Downloading data from https://upload.wikimedia.org/wikipedia/commons/thumb/7/7e/Rock-pap
er-scissors_%28rock%29.png/600px-Rock-paper-scissors_%28rock%29.png
228777/228777 [==============================] - 0s 0us/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 1s 803ms/step
1/1 [==============================] - 1s 634ms/step
Model pertama:
This image most likely belongs to Batu with a 57.61 percent confidence.
Model kedua:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model ketiga:
This image most likely belongs to Gunting with a 57.61 percent confidence.
Model keempat:
This image most likely belongs to Batu with a 57.61 percent confidence.
```



Kesimpulan

Pada akhirnya, ketika di test menggunakan data baru, tetap masih saja salah dalam prediksi padahal sudah menggunakan 4 model. Ini menunjukkan bahwa model yang cukup overfit. Menurut kami faktor utamanya adalah dari dataset yang dimana memang hanya sedikit, dan juga mungkin beragam faktor teknis di foto

tersebut, misal dari background foto, warna kulit tangan, angle foto, dll. Mungkin jika kita agak melebarkan dataset lagi atau istilahnya menambah data lagi, dengan background transparan atau background yang sama, dll. Kita mungkin dapat menyempurnakan data kita lagi. Jadi menurut kami faktor utamanya disini adalah sumber data yang digunakan untuk training model.