

```

# Nama: Rendika Nurhartanto Suharto
# Email: rendikarendi96@gmail.com
# ID Dcoding : RENDIKA NURHARTANTO SUHARTO

```

```

In [4]: from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

In [5]: !pip install opencv-python matplotlib

Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: numpy<1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.23.0)
Requirement already satisfied: contourpy<0.1.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.2.0)
Requirement already satisfied: cycler<0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools<4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.45.1)
Requirement already satisfied: kiwisolver<1.3 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging<20.8 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pyparsing<2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil<2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.8.2)
Requirement already satisfied: zipp<1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil<2.7-matplotlib) (1.16.0)

In [6]: !pip install split-folders

Collecting split-folders
  Downloading split-folders-0.5.1-py3-none-any.whl (0.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1

```

```

In [10]: import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import splitfolders
import time
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import keras as np
from google.colab import files
import keras.utils as image
import shutil
from keras.preprocessing import image

In [11]: import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow import keras

print(f"the version of tensorflow that use in this session is " + tf.__version__)

the version of tensorflow that use in this session is 2.14.0

```

Load File

```

In [16]: #path joining version for other path - Banyaknya data yang dimiliki
paper_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/Data Original/paper'
print("Paper:", sum(len(files) for _, _, files in os.walk(paper_dir)))
rock_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/Data Original/rock'
print("Rock:", sum(len(files) for _, _, files in os.walk(rock_dir)))
print("Scissors:", sum(len(files) for _, _, files in os.walk(scissors_dir)))

main_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/Data Original/'
print(f"Total Data:", sum(len(files) for _, _, files in os.walk(main_dir)))

Paper: 1752
Rock: 728
Scissors: 750
Total Data: 2280

In [17]: # Definisi path untuk direktori data train dan validation
train_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/train'
validation_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/validation'
test_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/testing Images'

# Buat direktori train dan validation untuk menampung data yang di split
# os.makedirs(train_dir, exist_ok=True)
# os.makedirs(validation_dir, exist_ok=True)

In [ ]: # Fungsi untuk membagi data menjadi train dan validation
def split_data(source, train, validation, split_size):
    files = os.listdir(source)
    # random sample files, len(files) # Acak urutan file untuk memastikan pemilihan acak
    random_sample_files = len(files)
    train_files = files[int(len(files) * split_size)]
    validation_files = files[int(len(files) * split_size):]

    # os.makedirs(train, exist_ok=True) # Buat direktori train jika belum ada
    # os.makedirs(validation, exist_ok=True) # Buat direktori validation jika belum ada

    # for file in train_files:
    #     shutil.copy(os.path.join(source, file), os.path.join(train, file))

    # for file in validation_files:
    #     shutil.copy(os.path.join(source, file), os.path.join(validation, file))

# # Bagi data untuk setiap kategori
# split_data(paper_dir, train_dir + '/paper', validation_dir + '/paper', 0.6)
# split_data(rock_dir, train_dir + '/rock', validation_dir + '/rock', 0.6)
# split_data(scissors_dir, train_dir + '/scissors', validation_dir + '/scissors', 0.6)

In [18]: # Hitung total data train dan validation
total_train = sum(len(files) for _, _, files in os.walk(train_dir))
total_validation = sum(len(files) for _, _, files in os.walk(validation_dir))

print(f"Total Data Train: {total_train}")
print(f"Total Data Validation: {total_validation}")

Total Data Train: 1312
Total Data Validation: 876

```

Image Generator + Augmentation Image

```

In [20]: # Image augmentation
train_datagenerator = ImageDataGenerator(
    rescale=1./255,
    rotation_range=90, # random rotation antara 0-90 derajat
    width_shift_range=0.2, #prosentasi width untuk random shift
    height_shift_range=0.2, #prosentasi height untuk random shift
    horizontal_flip=True, #horizontal flip untuk posisi tangan horizontal
    vertical_flip=True,
    zoom_range = 0.2,
    fill_mode = 'nearest')

validation_datagenerator = ImageDataGenerator(
    rescale=1./255,
    rotation_range=90, # random rotation antara 0-90 derajat
    width_shift_range=0.2, #prosentasi width untuk random shift
    height_shift_range=0.2, #prosentasi height untuk random shift
    horizontal_flip=True, #horizontal flip untuk posisi tangan horizontal
    vertical_flip=True,
    zoom_range = 0.2,
    fill_mode = 'nearest')

Berikut adalah penjelasan singkat tentang parameter yang digunakan dalam ImageDataGenerator:
1. rescale: Mengubah nilai piksel menjadi rentang 0 hingga 1 dengan cara membagi setiap piksel dengan 255.
2. rotation_range: Mengatur rentang untuk rotasi acak gambar dalam derajat.
3. width_shift_range dan height_shift_range: Menggeser gambar secara horizontal dan vertikal dengan rentang yang ditentukan sebagai persentase lebar atau tinggi gambar.
4. horizontal_flip: Melakukan flip horizontal secara acak pada gambar. Berguna untuk kasus di mana orientasi objek tidak penting.
5. brightness_range: Mengatur rentang kecerahan acak pada gambar.
6. shear_mode: Mengatur rentang peregangan (shear) aksis pada gambar.
7. fill_mode: Strategi pengisian yang digunakan untuk mengisi piksel yang kosong setelah peregangan atau rotasi gambar.1.

In [21]: #Persiapan Data Training dan Validasi Generator
train_generator = train_datagenerator.flow_from_directory(
    train_dir, # direktori data latih
    target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 piksel
    class_mode='categorical',
    batch_size=6, # klasifikasi Multi-Class (>2)
    shuffle = True,
    seed = 7)

validation_generator = validation_datagenerator.flow_from_directory(
    validation_dir, # direktori data validasi
    target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 piksel
    class_mode='categorical', # Klasifikasi Multi-Class (>2)
    shuffle = True,
    subset = "training", # Training berarti data validasi dibagi sedemikian rupa, dan yang ini digunakan untuk menguji performa model dengan validasi
    seed = 7)

Found 1312 images belonging to 3 classes.
Found 876 images belonging to 3 classes.

In [ ]: print(len(train_generator)) # ini adalah jumlah banyaknya step per generator melakukan image augmentasi pada seluruh gambar yang kita punya. jadi hitungannya adalah (Jumlah data/batch
train_generator.next()[27] # dipastikan dengan hal ini, jadi image generator melakukan augmentasi step ke 27, dan diubah ke 29 maka akan error. jika classmode adalah categorical
27

Jadi, keduanya berhubungan dengan cara bahwa len(train_generator) memberikan jumlah batch data yang dihasilkan oleh generator, sementara train_generator.next() digunakan untuk mengakses batch data yang dihasilkan
oleh generator.

```

```

In [ ]: # train_generator.next().shape # jika menggunakan ini, class_mode harus 'none'

In [23]: # train_generator.next() = (38,185)
# for i in range(9):
#     plt.subplot(3,3, i + 1)
#     img = train_generator.next()[0][0]
#     plt.imshow(img)
#     plt.axis('off')
# plt.show()

Saving Image Generator Result

In [ ]: # Set the directory where you want to save the generated images for train and validation
train_save_directory = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/generated images/train/'
validation_save_directory = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/generated images/validation/'

# Make sure the directories exist or create them if they don't
# os.makedirs(train_save_directory, exist_ok=True)
# os.makedirs(validation_save_directory, exist_ok=True)

In [ ]: # Set the number of batches you want to generate and save for each generator
num_batches_to_generate = 25

# Function to save generated images for a specific generator
def save_generated_images(generator, save_directory):
    # for i in range(num_batches_to_generate):
    #     batch = generator.next() # Get the next batch of images and labels
    #     images = batch[0] # Extract images from the batch
    #     labels = batch[1] # Extract labels from the batch

    #     for j, image in enumerate(images):
    #         image_file_name = f'generated_image.{i + num_batches_to_generate + j}.png' # Create a unique filename for each image
    #         image_save_path = os.path.join(save_directory, image_file_name) # Path to save the image

    #         # Save the image to the specified path
    #         # Assuming the images are in the format of Numpy arrays
    #         # Modify the saving process based on the format of images you have
    #         # For example, using PIL: image.fromarray(image).save(image_save_path)
    #         # Or using OpenCV: cv2.imwrite(image_save_path, image)
    #         # Here we assume the images are in the range of 0-255 and need to be converted to uint8
    #         image = (image * 255).astype('uint8')
    #         # Save the image as PNG format
    #         image.fromarray(image).save(image_save_path)

# Save generated images for train generator
save_generated_images(train_generator, train_save_directory)

# Save generated images for validation generator
save_generated_images(validation_generator, validation_save_directory)

In [ ]: # path joining version for other paths - Banyaknya data yang dimiliki
# train_save_directory = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/generated images/train'
# validation_validation_dir = '/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/rockpaperscissors/generated images/validation'
# print("Augmentation Image for Train Set:", sum(len(files) for _, _, files in os.walk(augmentation_train_dir)))
# print("Augmentation Image for Validation Set:", sum(len(files) for _, _, files in os.walk(augmentation_validation_dir)))

```

Build Model ANN and Callback Function

Callback for monitoring the accuracy

```

In [ ]: #Penggunaan Callback untuk menghentikan proses training sesuai akurasi yang diinginkan

class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        val_accuracy = logs.get('val_accuracy')
        accuracy = logs.get('accuracy')

        if val_accuracy is not None and accuracy is not None:
            if val_accuracy > 0.90 and accuracy > 0.98:
                print(f"Validasi >90% nh akurasi juga sudah dihentikan saja!")
                self.model.stop_training = True
            else:
                print(f"Validasi akurasi tidak teresida. Lanjutkan training.")
        else:
            print(f"Validasi akurasi tidak teresida. Lanjutkan training.")

callbacks = [myCallback()]

```

Deep learning Arsitektur sequential model

```

In [ ]: #Model sederhana Deep Learning 3 Conv dan 2 fully Connected Layer (dense layer)
model = 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(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(1024), # melakukan dropout(memastikan neuron) pada 308 layer sebelumnya
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(512, activation='softmax') #3 kelas
])

In [ ]: #Meng-compile model dengan Adam Optimizer dengan loss Categorical_Crossentropy karena multiclass classification
model.compile(loss=tf.keras.losses.CategoricalCrossentropy,
              optimizer=tf.optimizers.Adam(),
              metrics=['accuracy'])

In [ ]: model.summary()

Model: "sequential_6"
Layer (type) Output Shape Param
=====
conv2d_24 (Conv2D) (None, 148, 148, 32) 896
max_pooling2d_24 (MaxPooli (None, 74, 74, 32) 0
n2D)
conv2d_25 (Conv2D) (None, 72, 72, 64) 18496
max_pooling2d_25 (MaxPooli (None, 36, 36, 64) 0
n2D)
conv2d_26 (Conv2D) (None, 34, 34, 64) 36928
max_pooling2d_26 (MaxPooli (None, 17, 17, 64) 0
n2D)
conv2d_27 (Conv2D) (None, 15, 15, 128) 73856
max_pooling2d_27 (MaxPooli (None, 7, 7, 128) 0
n2D)
flatten_6 (Flatten) (None, 6272) 0
dropout_6 (Dropout) (None, 6272) 0
dense_12 (Dense) (None, 512) 3211776
dense_13 (Dense) (None, 3) 1539

=====
Total params: 334341 (12.75 Mb)
Trainable params: 334341 (12.75 Mb)
Non-Trainable params: 0 (0.00 Byte)

```

Visualize of my model architecture

```

In [ ]: model_viz = tf.keras.utils.plot_model(model,
# to file= 'model.png',
show_shapes=True,
show_layer_names=True,
rankdir='TB',
expand_nested=True,
dpi=100)

Out [ ]:
conv2d_24_input input: [(None, 150, 150, 3)]
Input/output output: [(None, 150, 150, 3)]

conv2d_24 input: (None, 150, 150, 3)
conv2D output: (None, 148, 148, 32)

max_pooling2d_24 input: (None, 148, 148, 32)
MaxPooling2D output: (None, 74, 74, 32)

conv2d_25 input: (None, 74, 74, 32)
conv2D output: (None, 72, 72, 64)

max_pooling2d_25 input: (None, 72, 72, 64)
MaxPooling2D output: (None, 36, 36, 64)

conv2d_26 input: (None, 36, 36, 64)
conv2D output: (None, 34, 34, 64)

max_pooling2d_26 input: (None, 34, 34, 64)
MaxPooling2D output: (None, 17, 17, 64)

conv2d_27 input: (None, 17, 17, 64)
conv2D output: (None, 15, 15, 128)

max_pooling2d_27 input: (None, 15, 15, 128)
MaxPooling2D output: (None, 7, 7, 128)

flatten_6 input: (None, 6272)
Flatten output: (None, 6272)

dropout_6 input: (None, 6272)
Dropout output: (None, 6272)

dense_12 input: (None, 6272)
Dense output: (None, 512)

dense_13 input: (None, 512)
Dense output: (None, 3)

```

Let's Train the Model

```

In [ ]: # Jumlah data dalam train_generator
total_train_data = train_generator.n
print("Total data di train_generator:", total_train_data)

# Jumlah data dalam validation_generator
total_validation_data = validation_generator.n
print("Total data di validation_generator:", total_validation_data)

Total data di train_generator: 1312
Total data di validation_generator: 876

Assign the initial parameter values for Training Model

In [ ]: epoch = 100 # Tentukan banyaknya model melakukan satu putaran penuh terhadap seluruh dataset training.
batch_size = 50 # seberapa banyak model membagi data train pada setiap epoch yang dilakukan

# Hitung steps_per_epoch dan validation_steps
steps_per_epoch = int(np.ceil(total_train_data / batch_size)) # ini tergantung pada banyak data dan jumlah batch_sizanya
validation_steps = int(np.ceil(total_validation_data / batch_size)) # ini tergantung pada banyak data dan jumlah batch_sizanya

What happend to the model training??

this is happended ?? Uraian terpenjint tentang implikasi dari nilai-nilai yang telah ditentukan terhadap komponen-komponen seperti epochs, batch size, steps per epoch, dan validation steps dalam konteks
pelatihan model.

Nilai-nilai yang ditentukan:
• Epochs: 350
• Batch size: 4
• Total data training: 700
• Total data validasi: 700

Interpretasi:
1. Epochs (Epoch):
    • Dengan jumlah epochs yang cukup besar (350), model akan melihat dataset training sebanyak 350 kali selama proses pelatihan.
    • Penggunaan epochs yang tinggi dapat memberikan kesempatan bagi model untuk mempelajari pola yang lebih dalam dari data, tetapi harus dipertimbangkan untuk menghindari overfitting.
2. Batch size:
    • Dengan batch size yang kecil (4), model akan melihat dan memperbarui bobotnya setiap kali memiliki 4 sampel data.
    • Batch size kecil bisa menghasilkan konvergensi yang lebih baik namun bisa memperlambat proses pelatihan.
3. Steps per epoch:
    • Jumlah steps per epoch dihitung berdasarkan total data training dibagi dengan batch size. Dalam konteks ini, akan ada 175 iterasi (jumlah sampel data training dibagi dengan batch size) pada setiap epoch.
4. Validation steps:
    • Validation steps dihitung dengan cara yang serupa seperti steps per epoch, namun pada dataset validasi. Dalam hal ini, akan ada 175 iterasi (jumlah sampel data validasi dibagi dengan batch size) pada setiap
    epoch untuk mengevaluasi model terhadap data validasi.

Implikasi dan Hubungan:
    • Dengan 350 epochs dan batch size 4, akan ada 175 iterasi pada setiap epoch yang melibatkan seluruh dataset training.
    • Setiap iterasi akan menggunakan batch size 4 untuk memperbarui bobot model.
    • Proses evaluasi pada dataset validasi akan dilakukan 175 kali pada setiap epoch untuk mengukur performa model terhadap data validasi.

In [ ]: from tensorflow.keras.callbacks import EarlyStopping

# Tentukan kondisi EarlyStopping
early_stopping = EarlyStopping(
    monitor='val_accuracy', # Monitor akurasi pada data validasi (nilai ini yang diperhatikan)
    patience=5, # Toleransi untuk tidak ada peningkatan (10 iterasi jika akurasi menunjukkan nilai-nilai yang konstan pada step)
    verbose=1, # Menampilkan detail tentang kondisi EarlyStopping
    restore_best_weights=True, # Memulihkan bobot terbaik ketika berhenti
)

Parameter verbose

dalam objek EarlyStopping, mengontrol tingkat detail atau kejelasan pesan yang akan ditampilkan selama proses pelatihan ketika kondisi EarlyStopping terpenuhi.

Nilai-nilai yang dapat diberikan kepada parameter verbose adalah:
    • 0: Tidak ada pesan yang ditampilkan selama proses pelatihan berlangsung.
    • 1: Menampilkan pesan tentang penghentian secara otomatis, misalnya "Early stopping condition triggered".
    • 2: Menampilkan pesan yang lebih rinci, memberikan informasi tentang kondisi penghentian dan langkah-langkah yang diambil saat kriteria EarlyStopping terpenuhi.

In [ ]: start_time = time.time()
# history model
train_generator, # Data generator untuk data latih
steps_per_epoch=steps_per_epoch, # Jumlah iterasi dalam setiap epoch (dihitung dari total data latih dan batch size)
validation_data=validation_data, # Data generator untuk data validasi
early_stopping=early_stopping, # Kondisi penghentian
callbacks=[callbacks, early_stopping], # Daftar callbacks yang akan digunakan selama pelatihan (informasi lebih detail)
verbose=1 # Level kejelasan pesan yang akan ditampilkan selama pelatihan (informasi lebih detail)

)

Epoch 1/100
27/27 [=====] - ETA: 8s - loss: 1.0872 - accuracy: 0.3742
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s/step - loss: 1.0872 - accuracy: 0.3742 - val_loss: 0.1093 - val_accuracy: 0.4235
Epoch 6/100
27/27 [=====] - ETA: 8s - loss: 1.0489 - accuracy: 0.4444
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s - loss: 0.9418 - accuracy: 0.5290
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s/step - loss: 0.9418 - accuracy: 0.5290 - val_loss: 0.6712 - val_accuracy: 0.7363
Epoch 47/100
27/27 [=====] - ETA: 8s - loss: 0.6961 - accuracy: 0.7180
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s - loss: 0.6961 - accuracy: 0.7180 - val_loss: 0.4665 - val_accuracy: 0.8858
Epoch 67/100
27/27 [=====] - ETA: 8s - loss: 0.5994 - accuracy: 0.8041
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s - loss: 0.5994 - accuracy: 0.8041 - val_loss: 0.3139 - val_accuracy: 0.9178
Epoch 67/100
27/27 [=====] - ETA: 8s - loss: 0.3464 - accuracy: 0.8918
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s/step - loss: 0.3464 - accuracy: 0.8918 - val_loss: 0.2283 - val_accuracy: 0.9486
Epoch 71/100
27/27 [=====] - ETA: 8s - loss: 0.3305 - accuracy: 0.8992
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s/step - loss: 0.3305 - accuracy: 0.8992 - val_loss: 0.2453 - val_accuracy: 0.9429
Epoch 87/100
27/27 [=====] - ETA: 8s - loss: 0.2389 - accuracy: 0.9177
Belum mencapai akurasi >90%, lanjutkan training.
27/27 [=====] - ETA: 8s/step - loss: 0.2389 - accuracy: 0.9177 - val_loss: 0.2459 - val_accuracy: 0.9418
-- Waktu training adalah 12.74793755234166 menit --

```

What happened above:

```

Proses pelatihan dilakukan dengan menggunakan model.fit() dengan generator data untuk sampel latih dan validasi. Jumlah iterasi yang dilakukan dalam setiap epoch diatur oleh
steps_per_epoch dan validation_steps yang dihitung berdasarkan ukuran batch dan total sampel data. Callbacks seperti callbacks dan early_stopping digunakan untuk mengontrol dan memantau
pelatihan model, seperti penghentian jika kriteria terpenuhi. Informasi tentang kemajuan pelatihan akan ditampilkan pada tingkat kejelasan (verbose) yang telah ditentukan. Setelah pelatihan selesai,
jika total yang diperlukan untuk pelatihan akan dicetak sebagai informasi tambahan.

Evaluasi Model

In [ ]: plt.plot(history.history['accuracy'], label='Training', color='blue')
plt.plot(history.history['val_accuracy'], label='Validation', color='red')
plt.title('Accuracy Training & Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['lower right'])
plt.show()

Accuracy Training & Validation

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Epoch

Training
Validation

In [ ]: plt.plot(history.history['loss'], label='Training', color='blue')
plt.plot(history.history['val_loss'], label='Validation', color='red')
plt.title('Loss Training & Validation')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['upper right'])
plt.show()

Loss Training & Validation

1.0
0.8
0.6
0.4
0.2
0

Epoch

Training
Validation

In [ ]: # Melakukan evaluasi terhadap model menggunakan data validasi
model.evaluate(validation_generator, steps=validation_steps, verbose=1)

# Menampilkan hasil evaluasi
print(f"Loss: {evaluation[0]}")
print(f"Accuracy: {evaluation[1]}")

Loss: 0.245885554243835
Accuracy: 0.94188855877686

So Final Accuracy here is 94%

Save the model

In [ ]: # 'model' akan di save ke local file
model.save('/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/RockPaperScissors_model12') # Replace 'path_to_save_model' with your desired path

In [5]: from tensorflow.keras.models import load_model

# Load the saved model
loaded_model = load_model('/content/drive/MyDrive/Colab Notebooks/Dicoding - IDCamp/Machine Learning/RockPaperScissors_model12') # Replace 'path_to_save_model' with the saved model

Try to predict here

In [15]: from keras.preprocessing import image
import numpy as np
from google.colab import files
import matplotlib.pyplot as plt

class_names = ['Kertas', 'Batik', 'Gunting'] # List nama kelas

# Fungsi untuk memprediksi gambar yang diunggah
def predict_uploaded_image(file_path, model):
    img = image.load_img(file_path, target_size=(150, 150))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    classes = model.predict(images, batch_size=50)
    return classes

# Memeriksa file gambar yang diunggah
uploaded = files.upload()

# Lakukan prediksi pada setiap file yang diunggah
for file_name in uploaded.keys():
    file_path = file_name
    result = predict_uploaded_image(file_path, loaded_model)
    predicted_class_index = np.argmax(result)
    predicted_class_name = class_names[predicted_class_index] # Sesuaikan dengan nama kelas Anda
    print(f"File: {file_name}, Predicted Class: {predicted_class_name}")

# Tampilkan gambar
img = image.load_img(file_path, target_size=(150, 150))
plt.imshow(img)
plt.title(f'Predicted Class: {predicted_class_name}')
plt.axis('off') # Hilangkan sumbu x dan y
plt.show()

Choose File: No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving paper_7.jpg to paper_7.jpg
1/4 [=====] - ETA: 0s - 38ms/step
File: paper_7.jpg, Predicted Class: Kertas

Predicted Class: Kertas

```