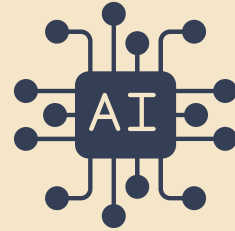


KECERDASAN ARTIFISIAL

IMPLEMENTASI JARINGAN SARAF TIRUAN

➤ **M. ILHAAM GHIFFARI (2108107010005)**



KECERDASAN ARTIFISIAL

TABLE OF CONTENT

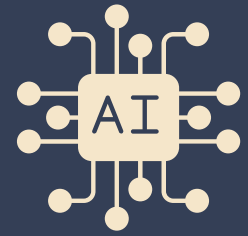
DATASET

PRE-PROCESS

MODEL SARAF TIRUAN

PROSES TRAINING

HASIL TRAINING



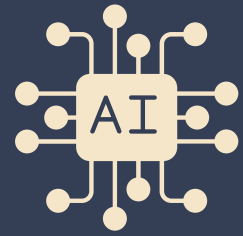
DATASET



ROCK, PAPER, SCISSORS DATASET

Dataset Ini mencakup banyak koleksi gambar yang disegmentasi menjadi tiga kategori: batu, kertas, dan gunting. Setiap kategori berisi gambar tangan dalam posisi yang mewakili gerakan masing-masing. Kumpulan datanya seimbang, memastikan keterwakilan yang setara untuk setiap kategori, yang sangat penting untuk melatih model yang tidak bias.

<https://www.kaggle.com/drgfreeman/rockpaperscissors>



PRE-PROCESS

MELAKUKAN SPLITTING

Dengan menggunakan modul `train_test_split` dari Scikit-Learn, kode membagi file citra dalam setiap kategori menjadi dua bagian, yaitu data pelatihan (`train_files`) dan data validasi (`val_files`), dengan proporsi 60% untuk pelatihan dan 40% untuk validasi. Hasilnya adalah dataset yang siap digunakan untuk melatih dan menguji model jaringan saraf guna tugas pengenalan gambar seperti rock-paper-scissors. Proses ini memungkinkan evaluasi kinerja model saat pelatihan dan menghindari overfitting.

```
1  from sklearn.model_selection import train_test_split
2  import shutil
3
4  base_dir = '/tmp/rockpaperscissors/rps-cv-images'
5  train_dir = os.path.join(base_dir, 'train')
6  validation_dir = os.path.join(base_dir, 'val')
7
8  if not os.path.exists(train_dir):
9      os.mkdir(train_dir)
10 if not os.path.exists(validation_dir):
11     os.mkdir(validation_dir)
12
13 for category in ['rock', 'paper', 'scissors']:
14     category_dir = os.path.join(base_dir, category)
15     train_category_dir = os.path.join(train_dir, category)
16     val_category_dir = os.path.join(validation_dir, category)
17
18     if not os.path.exists(train_category_dir):
19         os.mkdir(train_category_dir)
20     if not os.path.exists(val_category_dir):
21         os.mkdir(val_category_dir)
22
23     train_files, val_files = train_test_split(os.listdir(category_dir), test_size=0.4)
24
25     for file in train_files:
26         shutil.copy(os.path.join(category_dir, file), train_category_dir)
27     for file in val_files:
28         shutil.copy(os.path.join(category_dir, file), val_category_dir)
```

PRE-PROCESS

IMAGE DATA GENERATOR

Kode ini menggunakan ImageDataGenerator dari TensorFlow Keras untuk meningkatkan dataset gambar pelatihan dengan berbagai transformasi seperti rotasi, pergeseran, dan pemutaran.

Ini membantu mencegah overfitting dan meningkatkan kemampuan model dalam mengenali variasi dalam data gambar. Selain itu, kode ini mempersiapkan batch data dari direktori pelatihan dan validasi untuk digunakan dalam pelatihan dan pengujian model jaringan saraf.

```
1 from tensorflow.keras.preprocessing.image import ImageDataGenerator
2
3 train_datagen = ImageDataGenerator(
4     rescale=1./255,
5     rotation_range=60,
6     width_shift_range=0.3,
7     height_shift_range=0.3,
8     shear_range=0.3,
9     zoom_range=0.3,
10    horizontal_flip=True,
11    vertical_flip=True,
12    fill_mode='reflect'
13 )
14
15 test_datagen = ImageDataGenerator(rescale=1./255)
16
17 train_generator = train_datagen.flow_from_directory(
18     train_dir,
19     target_size=(150, 150),
20     batch_size=32,
21     class_mode='categorical'
22 )
23
24 validation_generator = test_datagen.flow_from_directory(
25     validation_dir,
26     target_size=(150, 150),
27     batch_size=32,
28     class_mode='categorical'
29 )
30
```


MODEL



```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout
3 from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout
4
5
6 model = Sequential([
7     Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
8     MaxPooling2D(2, 2),
9     Conv2D(128, (3,3), activation='relu'),
10    MaxPooling2D(2,2),
11    Conv2D(256, (3,3), activation='relu'),
12    MaxPooling2D(2,2),
13    Conv2D(512, (3,3), activation='relu'),
14    MaxPooling2D(2,2),
15    GlobalAveragePooling2D(),
16    Dense(1024, activation='relu'),
17    Dropout(0.5),
18    Dense(3, activation='softmax')
19 ])
20
21 model.compile(optimizer='adam',
22               loss='categorical_crossentropy',
23               metrics=['accuracy'])
24
```

- Jenis jaringan yang digunakan adalah Convolutional Neural Network (CNN), yang cocok untuk tugas pengenalan gambar seperti yang terlihat dalam penggunaan lapisan Conv2D, MaxPooling2D, dan GlobalAveragePooling2D.
- Algoritma optimisasi yang digunakan adalah Adam (Adaptive Moment Estimation). Hal ini terlihat dalam baris kode **model.compile(optimizer='adam', ...)**.
- Fungsi aktivasi yang digunakan untuk sebagian besar lapisan adalah ReLU (Rectified Linear Unit) dengan pengecualian lapisan output yang menggunakan softmax. Fungsi aktivasi ReLU digunakan dalam lapisan-lapisan Conv2D dan Dense untuk memperkenalkan non-linearitas ke dalam jaringan, sedangkan softmax digunakan untuk menghasilkan distribusi probabilitas pada lapisan output.

MODEL

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 148, 148, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 74, 74, 64)	0
conv2d_1 (Conv2D)	(None, 72, 72, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 128)	0
conv2d_2 (Conv2D)	(None, 34, 34, 256)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 256)	0
conv2d_3 (Conv2D)	(None, 15, 15, 512)	1180160
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 1024)	525312
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 3)	3075
=====		
Total params: 2079363 (7.93 MB)		
Trainable params: 2079363 (7.93 MB)		
Non-trainable params: 0 (0.00 Byte)		
=====		

- Ada empat lapisan konvolusi (Conv2D) dalam jaringan, dengan ukuran filter yang berbeda pada setiap lapisan.
- Setelah setiap lapisan konvolusi, ada lapisan MaxPooling2D yang mengurangi dimensi data dengan faktor 2.
- Lapisan global_average_pooling2d diikuti oleh dua lapisan Dense (fully connected) yang terakhir digunakan untuk menghasilkan output kelas dengan aktivasi softmax.
- Jumlah total node dalam hidden layer adalah $64 + 128 + 256 + 512 = 960$
- Total Weight dalam jaringan ini adalah 2,079,363 (sekitar 7.93 MB)

TRAINING

CALLBACKS




```
1 from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, TensorBoard
2 import datetime
3
4 log_dir = "/content/logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
5 tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
6
7
8 early_stopping = EarlyStopping(monitor='val_loss', patience=5)
9 model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)
10 reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=0.00001)
11
12 callbacks = [early_stopping, model_checkpoint, reduce_lr, tensorboard_callback]
```

TensorBoard digunakan untuk visualisasi proses pelatihan, EarlyStopping menghentikan pelatihan jika tidak ada peningkatan pada metrik yang dipantau (dalam hal ini val_loss) setelah beberapa epoch (ditetapkan dengan patience=5), ModelCheckpoint menyimpan model pada saat performanya terbaik berdasarkan val_loss, dan ReduceLROnPlateau mengurangi learning rate ketika tidak ada peningkatan pada val_loss setelah beberapa epoch (ditetapkan dengan patience=3) dengan faktor pengurangan tertentu dan batas learning rate minimum.

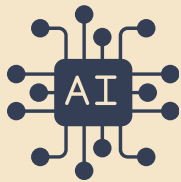
TRAINING

FIT



```
1 history = model.fit(  
2     train_generator,  
3     steps_per_epoch=25,  
4     epochs=25,  
5     validation_data=validation_generator,  
6     validation_steps=5,  
7     verbose=2,  
8     callbacks=callbacks  
9 )  
10
```

Kode tersebut menjalankan proses pelatihan (fit) sebuah model pembelajaran mesin dengan menggunakan data dari `train_generator` dan `validation_generator`. Setiap epoch akan melalui 25 langkah dengan data pelatihan dan 5 langkah dengan data validasi, untuk total 25 epoch. Informasi pelatihan akan ditampilkan dengan sedikit rincian (`verbose=2`) dan beberapa fungsi callback akan digunakan untuk memantau pelatihan,

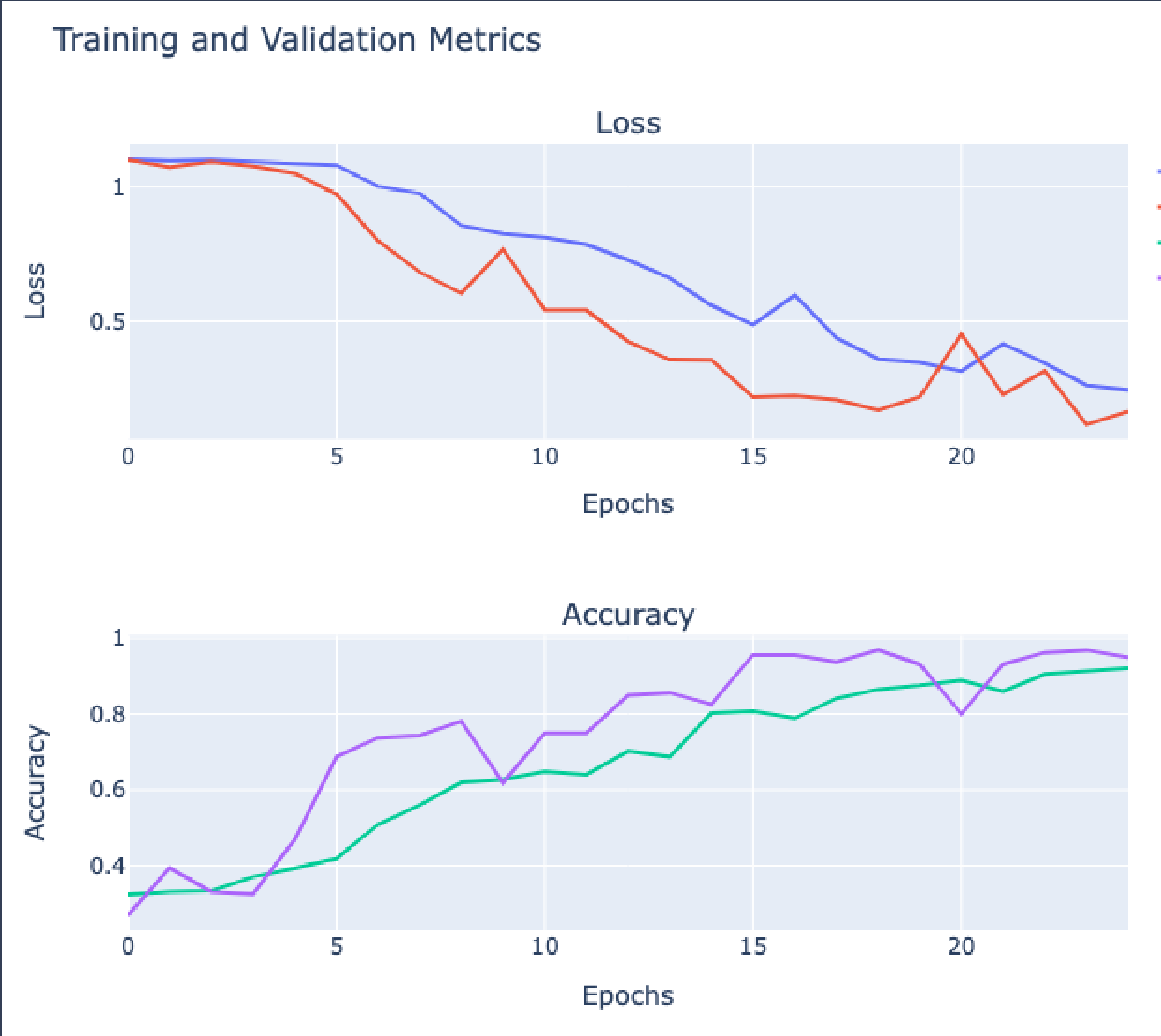


HASIL

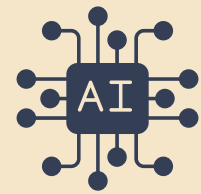
28/28 [=====] -
Akurasi: 96.12%

Grafik pertama menunjukkan "Loss" dan memiliki dua garis: "Train Loss" (biru) dan "Validation Loss" (merah). "Train Loss" menurun secara konsisten, yang menunjukkan model tersebut semakin baik dalam memprediksi atau mengklasifikasi data pelatihan seiring berjalannya waktu. "Validation Loss" menurun juga tetapi menunjukkan beberapa fluktuasi

Grafik kedua menunjukkan "Accuracy" dengan dua garis: "Train Accuracy" (hijau) dan "Validation Accuracy" (ungu). Keduanya meningkat seiring dengan waktu, yang merupakan indikator bahwa model tersebut memperbaiki kemampuannya dalam mengklasifikasikan baik data pelatihan maupun validasi.



- Train Loss
- Validation Loss
- Train Accuracy
- Validation Accuracy



THANK YOU

