Presentation By Rifqi arrayan

Rock, Paper, Scissors Image Classification Using CNN (Convolutional Neural Networks)

TOOLS

Google colab is used as platform for analyzing data

pyhton is used data for data analysis processes

open-source for building and training machine learning models, including neural networks.

Keras is a library that runs on top of TensorFlow, providing a simple interface for building and training models.

Background

- The main challenge in image classification is recognizing objects from different perspectives.
- With the development of technology, deep learning models can be used to automatically classify visual objects.

Why Use CNN?

- CNN is very efficient for image recognition tasks.
- CNN can learn from patterns in images such as edges, shapes, and more complex patterns.

Purpose

- Building a deep learning model that can recognize images of rock, scissors, and paper.
- Training the model to achieve high accuracy in classifying these images.
- Using early stopping techniques to avoid overfitting.

Dataset

- The dataset contains 2188 images divided into 3 classes: rock, scissors, and paper.
- The data is divided into 60% for training and 40% for validation.

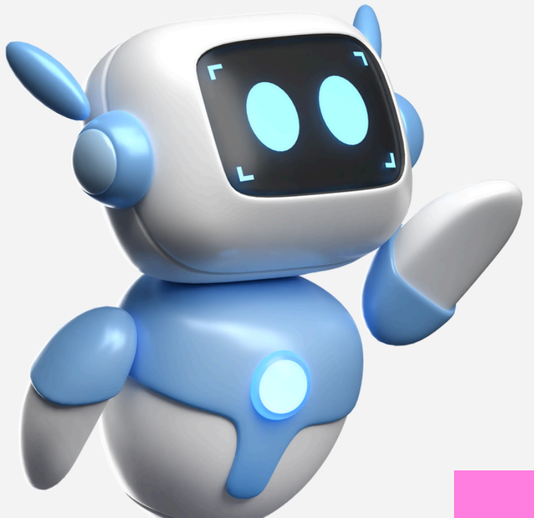
```
train_generator = train_datagen.flow_from_directory(
    dataset_folder,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='training'  # 60% data untuk training
)

validation_generator = train_datagen.flow_from_directory(
    dataset_folder,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation'  # 40% data untuk validation
)
```

- This images were resized to 150x150 pixels and captured in batches of size 32.
- Images are divided into training and validation sets with a ratio of 60:40.

Model Architecture

```
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(128, (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.Dropout(0.5),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')  # 3 classes (rock, paper, scissors)
])
```



4 Convolutional layers and Max Pooling.
Uses Dense layers with Dropout to reduce overfitting.

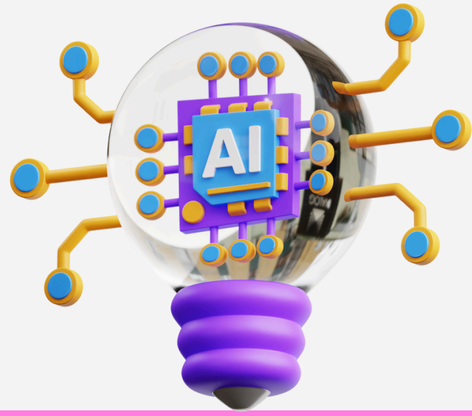
CNN model with multiple convolution and pooling layers to extract features from images.
Dropout layer with rate 0.5 to prevent overfitting.
Output using softmax for 3-class classification (rock, scissors, paper).

Model Compilation

```
# Compile model
model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
)
```

Model Compilation:
Using categorical_crossentropy loss since this is a multi-class classification.
Adam optimizer is used to speed up convergence.

The categorical_crossentropy loss function is suitable for classification problems with more than two classes.
Adam optimizer is popularly used because it is efficient in minimizing loss.



Callback dan Early Stopping

```python
# Custom callback untuk menghentikan pelatihan jika akurasi validasi sudah > 85%
class StopTrainingAtAccuracy(tf.keras.callbacks.Callback):
    def __init__(self, target_accuracy=0.85):
        super(StopTrainingAtAccuracy, self).__init__()
        self.target_accuracy = target_accuracy

    def on_epoch_end(self, epoch, logs=None):
        val_acc = logs.get("val_accuracy")
        if val_acc >= self.target_accuracy:
            print(f"\nAkurasi validasi mencapai {val_acc*100:.2f}% pada epoch {epoch+1}, menghentikan pelatihan...")
            self.model.stop_training = True

# Buat instance dari callback
stop_at_85_callback = StopTrainingAtAccuracy(target_accuracy=0.85)

# Train model dengan early stopping agar tidak overfitting dan callback penghentian otomatis
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)

# Latih model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    epochs=20,  # Batasi ke 20 epoch agar waktu training tidak melebihi 30 menit
    callbacks=[early_stop, stop_at_85_callback]  # Callback untuk menghentikan pelatihan otomatis saat akurasi lebih
)
```

Custom Callback and Early Stopping:
Training is stopped automatically when validation accuracy exceeds 85%

- Uses a callback to stop training if validation accuracy exceeds 85% to avoid overfitting.
- EarlyStopping also stops training if there is no improvement in validation loss for 3 consecutive epochs..

Model Evaluation

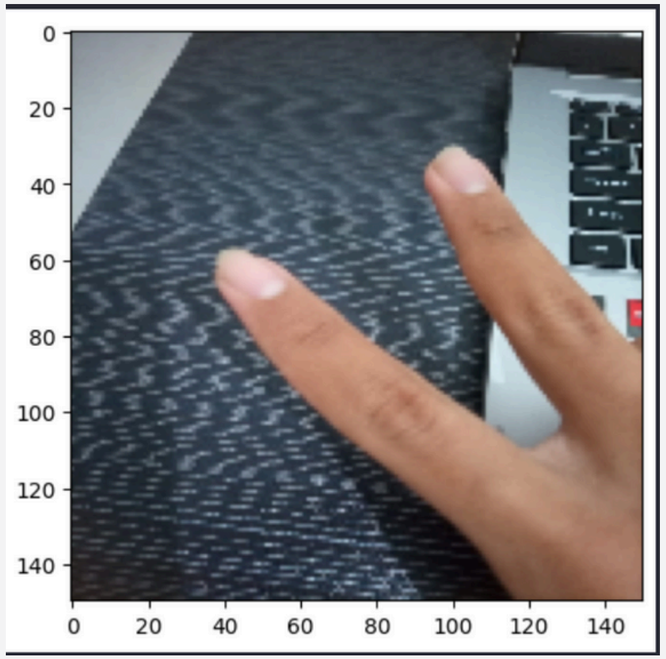
```python
# Evaluasi model
acc = model.evaluate(validation_generator)[1] * 100
print(f"Akurasi validation: {acc:.2f}%")
```

```
Akurasi validation: 86.38%
```

The model was evaluated on the validation set and achieved an accuracy of more than 85%.

The evaluation results show the accuracy of the model on the validation set.

Image predictions




```
print(fn)
if classes[0, 0] != 0:
    print('Gunting')
elif classes[0, 1] != 0:
    print('Batu')
else:
    print('Kertas')
```

```
WhatsApp Image 2024-09-19 at 13.33.14.jpeg
Gunting
```

Uploading an image and converting it to an array before being predicted by the model

and the model predicts that the image is a picture of scissors

Conclusion

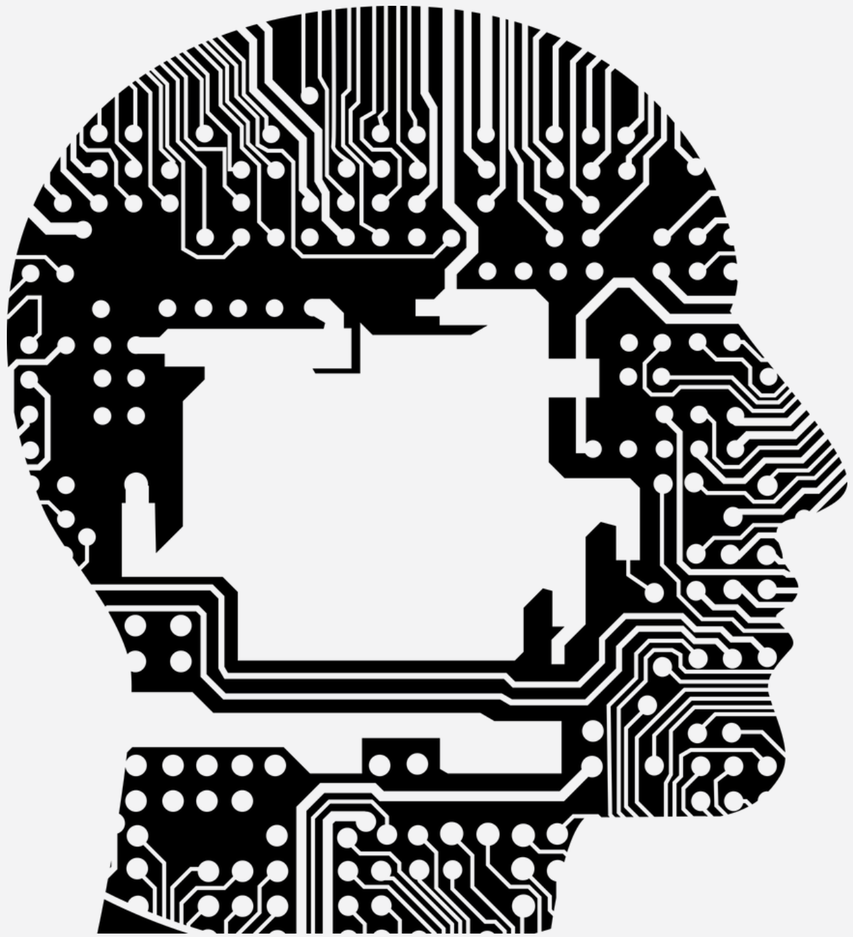
In this project, I successfully developed a Convolutional Neural Network (CNN) model that can classify images of rock, scissors, and paper with high accuracy. This process involved processing the dataset, building a CNN model, and evaluating and predicting new images. By implementing optimization techniques, including a custom callback to stop training when the validation accuracy reaches the target, I managed to achieve a validation accuracy above 85%.

This project demonstrates the power of deep learning, especially CNN, in image classification. CNN effectively extracts important features from images, while data augmentation techniques such as rotation, zoom, and flipping prevent overfitting and improve model generalization.

The implementation of custom callbacks and `EarlyStopping` helped me optimize training time and computing resources. In addition, this project opens up opportunities for the application of this technology in various fields, such as object recognition in real-time applications and automated systems.

Overall, I found that deep learning technology has great potential in visual pattern recognition, and the resulting models are not only efficient and accurate, but also flexible for future applications.

VIEW IN GITHUB AND VIEW SOURCE CODE



Thank You!