Telkom University

UAS MACHINE LEARNING

IMAGE CLASSIFICATION

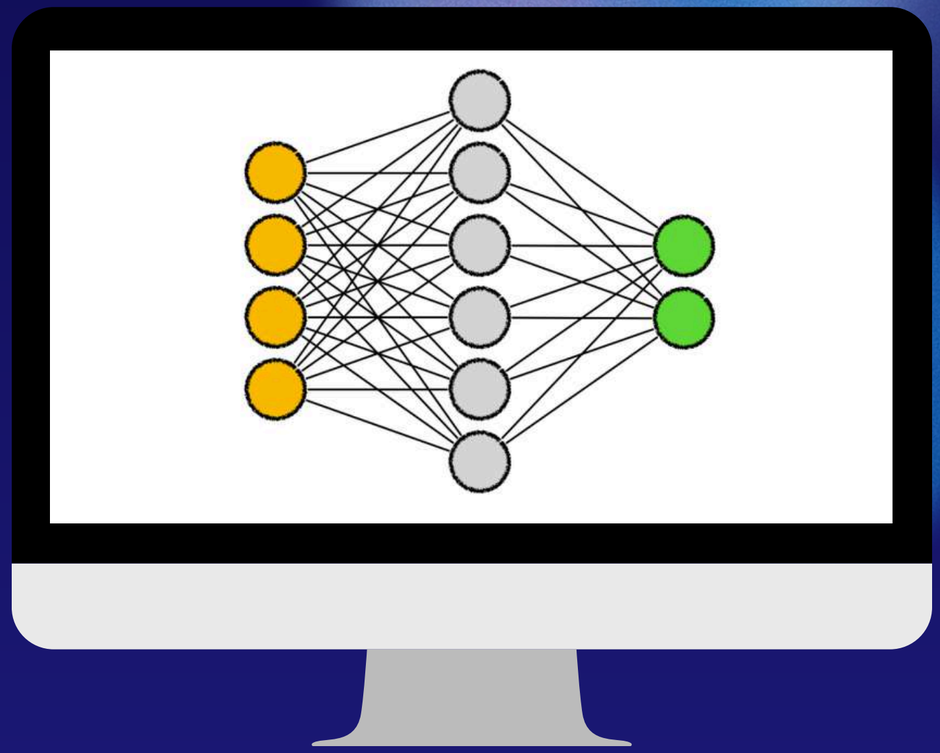
By:
Muhammad Raihan Butar-Butar
1103213077

PENDAHULUAN

Proyek ini menggunakan salah satu model deep learning yang populer bernama CNN (Convolutional Neural Network) untuk melakukan klasifikasi gambar. Dataset yang digunakan dalam proyek ini adalah MNIST yang berisi kumpulan gambar angka tulisan tangan.

CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN sering digunakan untuk klasifikasi gambar karena model ini dapat mempelajari fitur hierarki seperti tepi, tekstur, dan bentuk, sehingga memungkinkan pengenalan objek secara akurat dalam gambar. CNN unggul dalam tugas ini karena model ini dapat secara otomatis mengekstraksi fitur spasial yang bermakna dari gambar.



MNIST DATASET

MNIST (Modified National Institute of Standards and Technology) merupakan dataset yang umum digunakan untuk melatih berbagai sistem pemrosesan gambar. Dataset ini berisi 60.000 gambar pelatihan dan 10.000 gambar pengujian, yang masing-masing merupakan gambar skala abu-abu berukuran 28 × 28 pixel.

TUJUAN

Tujuan utama dari proyek ini adalah membangun sebuah model untuk melakukan klasifikasi gambar dengan tingkat akurasi minimal 90%.

WORKFLOW

IMPORT LIBRARY

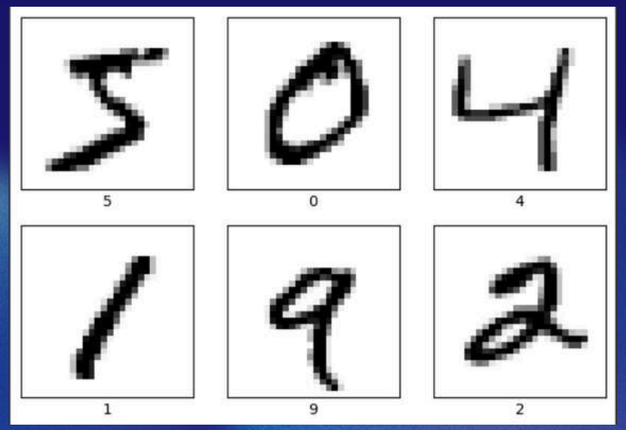
Sebelum memulai, pastikan library tensorflow sudah terinstall kemudian import libary tensorflow, numpy, dan matplotlib.

```
[ ]  # !pip install tensorflow
```

```
[ ]  import tensorflow as tf
     from tensorflow.keras import layers, models
     from tensorflow.keras.datasets import mnist
     import numpy as np
     import matplotlib.pyplot as plt
```

LOAD DATASET

Memuat dataset lalu bagi menjadi data latih dan data uji. Kemudian tampilkan beberapa gambar dari dataset jika diperlukan.




```
[ ]  # Load the MNIST dataset
     (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
[ ]  # Function to display a grid of images
     def plot_images(images, labels, num_images=12, num_cols=3):
         num_rows = num_images // num_cols
         plt.figure(figsize=(num_cols * 2, num_rows * 2))
         for i in range(num_images):
             plt.subplot(num_rows, num_cols, i + 1)
             plt.xticks([])
             plt.yticks([])
             plt.grid(False)
             plt.imshow(images[i], cmap=plt.cm.binary)
             plt.xlabel(labels[i])
         plt.tight_layout()
         plt.show()

     # Display the first 12 images from the training dataset
     plot_images(x_train, y_train, num_images=12, num_cols=3)
```

PREPROCESSING

Lakukan preprocessing data. Preprocessing dilakukan dalam beberapa tahap, yaitu:

1. Melakukan normalisasi gambar dengan membagi nilai piksel dengan 255 sehingga rentang nilainya menjadi [0,1]
2. Menambahkan dimensi ekstra pada x_train dan x_test untuk menyesuaikan bentuk input model yaitu 28x28x1
3. Mengubah label kelas y_train dan y_test dari representasi integer menjadi representasi one-hot encoded

```
[ ]  # Normalize the images to [0, 1] range
     x_train, x_test = x_train / 255.0, x_test / 255.0

     # Expand dimensions to match the input shape of the model (28x28x1)
     x_train = np.expand_dims(x_train, axis=-1)
     x_test = np.expand_dims(x_test, axis=-1)

     # Convert class vectors to binary class matrices (one-hot encoding)
     y_train = tf.keras.utils.to_categorical(y_train, 10)
     y_test = tf.keras.utils.to_categorical(y_test, 10)
```

MODELING

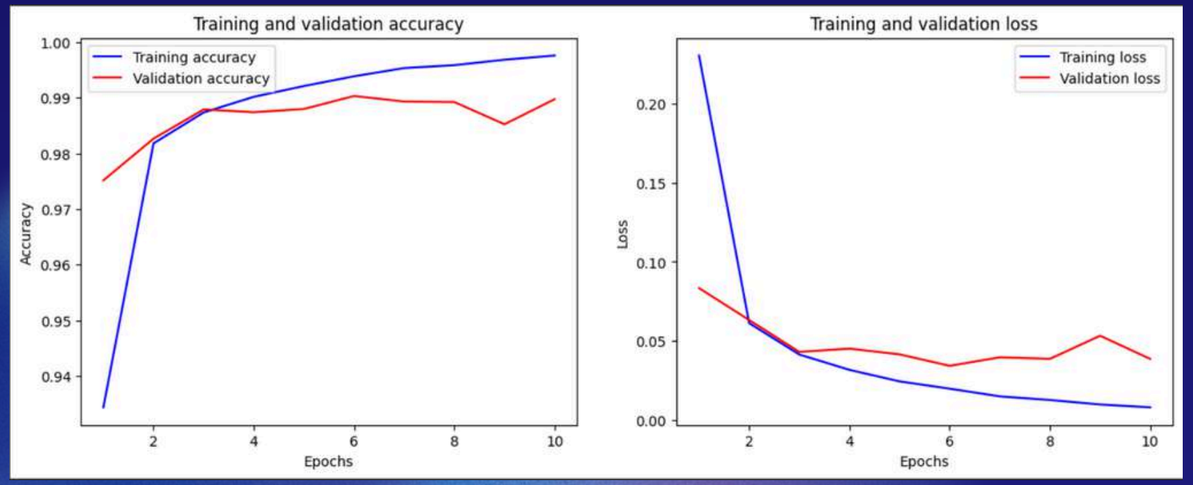
Membuat sebuah model Sequential yang merupakan tumpukan linier dari beberapa layer. Kemudian mengompilasi model dengan optimizer 'adam', yang merupakan algoritma optimasi yang efisien. Model ini dirancang untuk memproses gambar grayscale 28x28 pixel dan mengklasifikasikannya ke dalam salah satu dari 10 kelas.

```
[ ]  # Build the model
     model = models.Sequential([
         layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
         layers.MaxPooling2D(pool_size=(2, 2)),
         layers.Conv2D(64, (3, 3), activation='relu'),
         layers.MaxPooling2D(pool_size=(2, 2)),
         layers.Flatten(),
         layers.Dense(128, activation='relu'),
         layers.Dense(10, activation='softmax')
     ])

     # Compile the model
     model.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

TRAINING

Melatih model menggunakan data latih. Sebanyak 20% data latih digunakan untuk validasi. Setelah itu dilakukan visualisasi akurasi serta loss dari pelatihan dan validasi dalam bentuk line plot.




```
[ ]  # Train the model
     history = model.fit(x_train, y_train, epochs=10, batch_size=128, validation_split=0.2)
```

```
[ ]  def plot_history(history):
         acc = history.history['accuracy']
         val_acc = history.history['val_accuracy']
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         epochs = range(1, len(acc) + 1)

         plt.figure(figsize=(14, 5))

         plt.subplot(1, 2, 1)
         plt.plot(epochs, acc, 'b', label='Training accuracy')
         plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
         plt.title('Training and validation accuracy')
         plt.xlabel('Epochs')
         plt.ylabel('Accuracy')
         plt.legend()

         plt.subplot(1, 2, 2)
         plt.plot(epochs, loss, 'b', label='Training loss')
         plt.plot(epochs, val_loss, 'r', label='Validation loss')
         plt.title('Training and validation loss')
         plt.xlabel('Epochs')
         plt.ylabel('Loss')
         plt.legend()

         plt.show()

     # Plot the training history
     plot_history(history)
```

EVALUATE

Melakukan evaluasi model menggunakan data uji untuk mendapatkan nilai akurasi serta loss dari model.

```
[ ]  # Evaluate the model
     test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
     print(f'Test accuracy: {test_acc}')
```

```
313/313 - 1s - loss: 0.0316 - accuracy: 0.9910 - 846ms/epoch - 3ms/step
Test accuracy: 0.9909999966621399
```

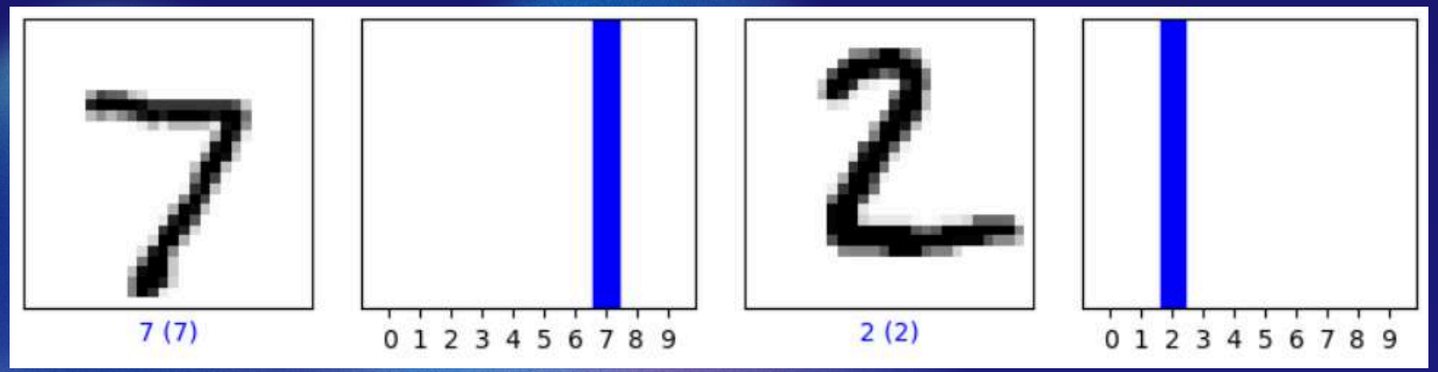
SAVE MODEL

Menyimpan model yang telah dilatih agar bisa digunakan kapan saja.

```
[ ]  # Save the model
     model.save('mnist_model.h5')
```

PREDICTING

Melakukan prediksi menggunakan data uji, kemudian hasil dari prediksi tersebut divisualisasikan dalam 2 bentuk. Fungsi plot_image untuk menampilkan gambar digit beserta prediksi dan nilai aslinya. Sedangkan fungsi plot_value_array menampilkan probabilitas prediksi untuk setiap kelas (0-9) dalam bentuk bar plot.




```
[ ]  # Load the model
     loaded_model = tf.keras.models.load_model('mnist_model.h5')

     # Predict on new data
     predictions = loaded_model.predict(x_test)

     # Display some predictions
     def plot_image(i, predictions_array, true_label, img):
         predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
         plt.grid(False)
         plt.xticks([])
         plt.yticks([])

         plt.imshow(img, cmap=plt.cm.binary)

         predicted_label = np.argmax(predictions_array)
         if predicted_label == np.argmax(true_label):
             color = 'blue'
         else:
             color = 'red'

         plt.xlabel(f"{predicted_label} ({np.argmax(true_label)})", color=color)

     def plot_value_array(i, predictions_array, true_label):
         predictions_array, true_label = predictions_array[i], true_label[i]
         plt.grid(False)
         plt.xticks(range(10))
         plt.yticks([])
         thisplot = plt.bar(range(10), predictions_array, color="#777777")
         plt.ylim([0, 1])
         predicted_label = np.argmax(predictions_array)

         thisplot[predicted_label].set_color('red')
         thisplot[np.argmax(true_label)].set_color('blue')

     num_rows = 5
     num_cols = 3
     num_images = num_rows * num_cols
     plt.figure(figsize=(2*2*num_cols, 2*num_rows))
     for i in range(num_images):
         plt.subplot(num_rows, 2*num_cols, 2*i+1)
         plot_image(i, predictions, y_test, x_test)
         plt.subplot(num_rows, 2*num_cols, 2*i+2)
         plot_value_array(i, predictions, y_test)
     plt.tight_layout()
     plt.show()
```

KESIMPULAN

Proyek ini berhasil membangun model CNN untuk melakukan klasifikasi gambar dengan tingkat akurasi sebesar ±99 %, dimana hal ini memenuhi tujuan dari proyek ini. Model ini sangat cocok untuk diaplikasikan ke berbagai sektor yang membutuhkannya dikarenakan akurasinya sangat tinggi yang hampir mencapai 100%.

THANK YOU FOR YOUR ATTENTION