VISI KOMPUTER DEEP LEARNING

by Kelompok 4

For Cooperation and
Collaboration Offer

April
2023

CONTENT

For Cooperation and Collaboration Offer

April
2023

ABOUT FACE RECOGNITION

Face recognition adalah teknologi yang mengidentifikasi atau memverifikasi identitas seseorang berdasarkan fitur wajah. Teknologi ini digunakan dalam berbagai bidang, seperti keamanan, pengawasan, otentikasi tanpa kontak, dan peningkatan pengalaman pengguna, misalnya dalam sistem pembayaran digital atau akses perangkat. Manfaatnya meliputi peningkatan keamanan, kemudahan verifikasi identitas, dan efisiensi operasional. Namun, tantangan terkait privasi dan etika juga perlu diperhatikan dalam penggunaannya.

For Cooperation and
Collaboration Offer

April
2023

DATASET YANG DIGUNAKAN

Dataset yang digunakan dalam penelitian ini adalah Dataset Celeb A. CelebA (CelebFaces Attributes Dataset) adalah kumpulan data yang terdiri dari lebih dari 200.000 gambar wajah dari sekitar 10.000 selebriti. Dataset ini sering digunakan untuk melatih dan menguji model pengenalan wajah serta berbagai tugas pengolahan citra lainnya.

Dataset yang digunakan dalam pengujian ini berjumlah sekitar 22.000 data citra, yang dipilih secara acak dari kumpulan CelebA. Jumlah label Pria dan Wanita (Male dan Female)

ARSITEKTUR DENSENET

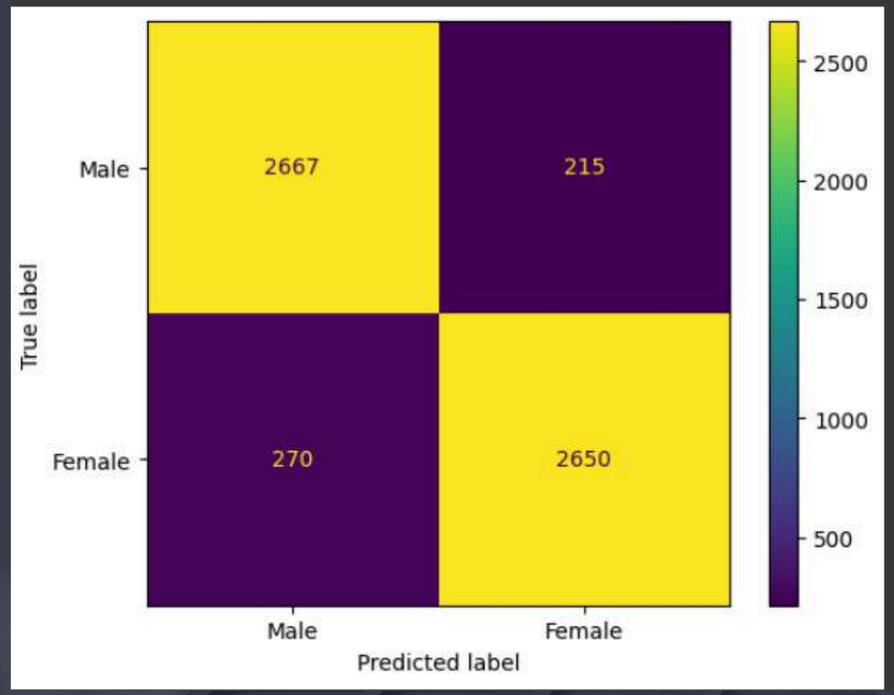
DenseNet201 adalah varian dari arsitektur jaringan saraf dalam yang dikenal sebagai DenseNet (Densely Connected Convolutional Networks), yang memperkenalkan konsep koneksi padat antara layer. Dalam DenseNet, setiap layer menerima input dari semua layer sebelumnya, bukan hanya dari layer terdekat, yang membantu meningkatkan efisiensi model dan mengatasi masalah vanishing gradient. Dengan 201 layer dan struktur dense block yang saling terhubung, DenseNet201 mampu menangkap fitur dengan baik sambil menggunakan lebih sedikit parameter dibandingkan dengan arsitektur lain yang lebih dalam. Karena keunggulannya, DenseNet201 sering digunakan dalam berbagai tugas computer vision, seperti pengenalan citra dan deteksi objek.

```
[ ]
    # Transfer Learning with DenseNet201
    base_densenet_model = tf.keras.applications.DenseNet201(weights='imagenet', include_top=False, input_shape=IMAGE_SIZE + (3,))

    # Unfreeze the last 5 layers of the base model
    for layer in base_densenet_model.layers[:-5]:  # Freeze all layers except the last 5
        layer.trainable = False
```

```
    # Create your full model
    densenet_model = Sequential([
        base_densenet_model,  # Base DenseNet201
        GlobalAveragePooling2D(),  # Global Average Pooling
        Dense(1024, activation='relu'),  # Fully connected layers
        Dropout(0.2),
        Dense(256, activation='relu'),
        Dropout(0.2),
        Dense(64, activation='relu'),
        Dropout(0.2),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(2, activation='softmax')  # Final classification layer (adjust output size as needed)
    ])

    # Compile the model
    base_learning_rate = 0.00001
    densenet_model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),  # Remove `from_logits` since softmax is used
        metrics=['accuracy']
    )
```

Model Arsitektur

ARSITEKTUR DENSENET



Confusion Matrix

```python
# Compile the model
base_learning_rate = 0.00001
densenet_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

```
Confusion Matrix:
        Male  Female
Male    2667     215
Female   270    2650

Classification Report:
              precision    recall  f1-score       support
Male           0.924956  0.907534  0.916162  2920.000000
Female         0.908069  0.925399  0.916652  2882.000000
accuracy       0.916408  0.916408  0.916408     0.916408
macro avg      0.916513  0.916467  0.916407  5802.000000
weighted avg   0.916568  0.916408  0.916406  5802.000000
```

Hyperparameter & Confusion Matrix

ARSITEKTUR RESNET

ResNet50V2 adalah varian dari arsitektur ResNet (Residual Network) yang dirancang untuk meningkatkan performa model dalam pengenalan citra dan tugas pengolahan citra lainnya. Dengan 50 lapisan, ResNet50V2 mengimplementasikan blok residual yang memungkinkan aliran informasi lebih baik melalui shortcut connections, yang menghubungkan output dari satu blok ke blok berikutnya. Versi kedua ini memperkenalkan beberapa peningkatan, termasuk urutan konvolusi yang berbeda dan penggunaan Batch Normalization yang lebih efisien, yang membantu mempercepat proses pelatihan dan meningkatkan akurasi model. ResNet50V2 juga dikenal karena kemampuannya untuk menangkap fitur kompleks dari data gambar sambil mempertahankan jumlah parameter yang relatif rendah, menjadikannya pilihan populer untuk aplikasi seperti klasifikasi citra, deteksi objek, dan segmentasi.

```
# Transfer Learning dengan ResNet50
base_resnet_model = tf.keras.applications.ResNet50V2(weights='imagenet', include_top=False, input_shape= IMAGE_SIZE + (3,))

for layer in base_resnet_model.layers[-5:]:  # Unfreeze 5 layer terakhir
    layer.trainable = False
```

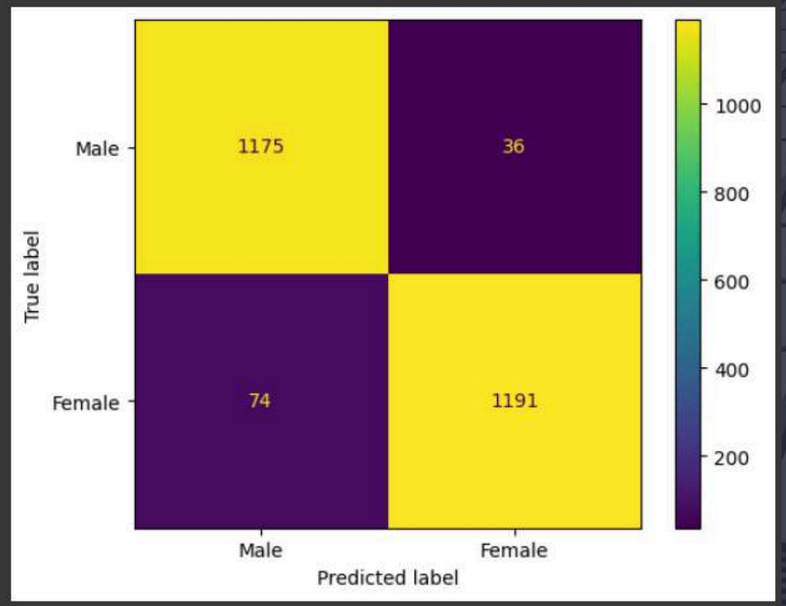
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels_notop.h5
94668760/94668760 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Membuat Arsitektur Fully Connected
resnet_model = Sequential([
    base_resnet_model,
    GlobalAveragePooling2D(),
    Dense(1024, activation='relu'),
    Dropout(0.2),
    Dense(256, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(2, activation="softmax")
])


# Kompile Model Deep Learning
base_learning_rate = 0.00001
resnet_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

Model Arsitektur

ARSITEKTUR RESNET

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=labels)
disp.plot()
plt.show()
```



Confusion Matrix

```
# Compile the model
base_learning_rate = 0.00001
densenet_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

Hyperparameter

```
Classification Report:
              precision    recall  f1-score   support
Male           0.974338  0.930435  0.951880  1265.000000
Female         0.930599  0.974401  0.951997  1211.000000
accuracy       0.951939  0.951939  0.951939     0.951939
macro avg      0.952469  0.952418  0.951939  2476.000000
weighted avg   0.952946  0.951939  0.951937  2476.000000
```

Confusion Matrix

ARSITEKTUR ALEXNET

AlexNet adalah arsitektur jaringan saraf dalam yang dikembangkan oleh Alex Krizhevsky, Ilya Sutskever, dan Geoffrey Hinton, dan terkenal karena menang dalam kompetisi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) pada tahun 2012. Dengan kedalaman 8 lapisan, termasuk 5 lapisan konvolusi diikuti oleh 3 lapisan fully connected, AlexNet merevolusi bidang computer vision dengan menunjukkan bahwa jaringan saraf dalam dapat mencapai akurasi yang jauh lebih tinggi dibandingkan metode tradisional. Arsitektur ini menggunakan teknik seperti ReLU (Rectified Linear Unit) sebagai fungsi aktivasi, pengurangan dimensi melalui max pooling, dan dropout untuk mengurangi overfitting. Selain itu, AlexNet memanfaatkan GPU untuk mempercepat pelatihan, yang menjadi salah satu faktor kunci kesuksesannya. Kontribusi AlexNet tidak hanya meningkatkan performa dalam pengenalan citra, tetapi juga menginspirasi pengembangan berbagai model jaringan saraf dalam lainnya yang lebih kompleks dan efisien.

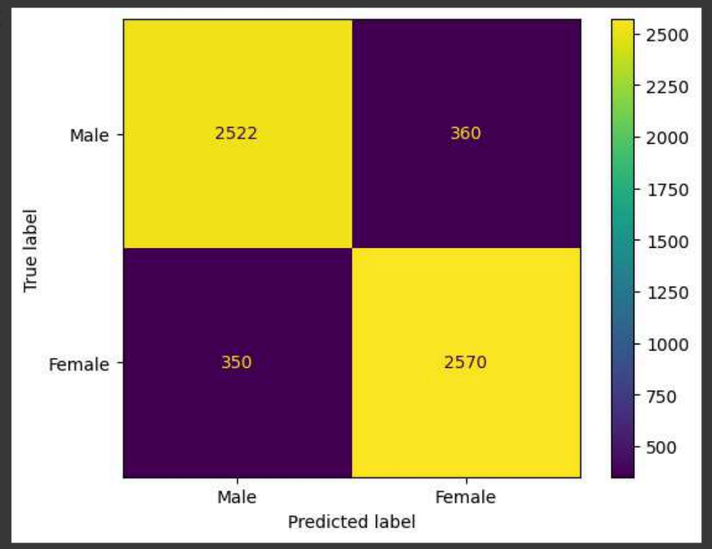
```
[ ] import os
    from tensorflow.keras import layers
    from tensorflow.keras import Model
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import tensorflow as tf
```

```
from keras.optimizers import Adam
model = tf.keras.models.Sequential([
    # 1st conv
  tf.keras.layers.Conv2D(96, (11,11),strides=(4,4), activation='relu', input_shape=(218, 178, 3)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2,2)),
    # 2nd conv
  tf.keras.layers.Conv2D(256, (11,11),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 3rd conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 4th conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 5th Conv
  tf.keras.layers.Conv2D(256, (3, 3), strides=(1, 1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2, 2)),
  # To Flatten layer
  tf.keras.layers.Flatten(),
  # To FC layer 1
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  #To FC layer 2
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(2, activation='sigmoid')
  ])


model.summary()
```

Model Arsitektur

ARSITEKTUR ALEXNET

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=labels)
disp.plot()
plt.show()
```



Confusion Matrix

```
base_learning_rate = 0.00001

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)

model.summary()
```

Hyperparameter

```
Confusion Matrix:
        Male  Female
Male    2522     360
Female   350    2570

Classification Report:
              precision    recall  f1-score      support
Male           0.877133  0.880137  0.878632  2920.000000
Female         0.878134  0.875087  0.876608  2882.000000
accuracy       0.877628  0.877628  0.877628     0.877628
macro avg      0.877633  0.877612  0.877620  5802.000000
weighted avg   0.877630  0.877628  0.877627  5802.000000
```

Confusion Matrix

GOOGLENET

GoogleNet ialah model yang diperkenalkan oleh google sejak 2014 dan menjadi peringkat pertama dalam kompetisi ILSVRC yang mendapatkan predikat sebagai arsitektur kinerja paling baik. Kelebihan googlenet terletak pada inception modules. Inception modules terdiri dari sejumlah convolution kecil yang digunakan untuk mereduksi.

ARSITEKTUR GOOGLENET

Arsitektur GoogLeNet adalah sebuah modifikasi arsitektur CNN yang berhasil menjadi model terbaik pada ILLSVRC14. Arsitektur ini bekerja dengan mendeteksi citra dengan lapisan yang dimiliki sejumlah lima hingga 22 lapisan tetapi tetap memiliki akurasi yang tinggi. Konsep kerja arsitektur ini didasarkan pada activation values pada deep network yang tidak sepenuhnya penting karena terdapat value of zero akibat korelasi sebelumnya, sehingga dibutuhkan activation values yang tidak terkoneksi sepenuhnya. Untuk memenuhi kondisi tersebut, pada GoogLeNet terdapat lapisan inception module yang terinspirasi dari model visual cortex manusia yang berperan untuk mengoptimalkan sparse structure sehingga menunjang komputasi.

```
# Transfer Learning dengan GoogleNet (InceptionV3)
base_inception_model = tf.keras.applications.InceptionV3(weights='imagenet', include_top=False, input_shape=IMAGE_SIZE + (3,))

for layer in base_inception_model.layers[-5:]:  # Unfreeze 5 layer terakhir
    layer.trainable = False
```

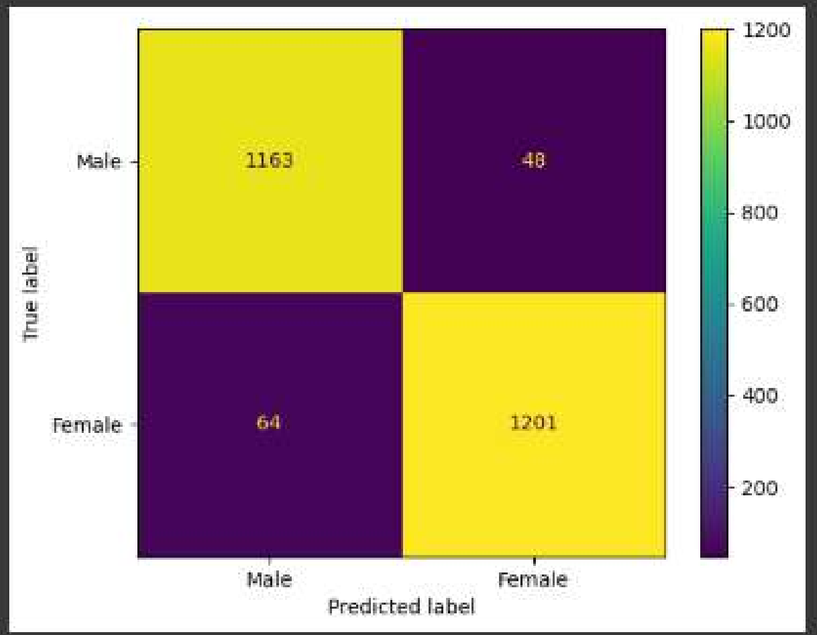
```
# Membuat arsitektur fully connected
inception_model = tf.keras.Sequential([
    base_inception_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(2, activation='softmax')  # Sesuaikan jumlah output sesuai dengan kebutuhan Anda
])

# Kompilasi model
base_learning_rate = 0.00001
inception_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

Model GoogleNet

April
2023

ARSITEKTUR GOOGLENET

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=labels)
disp.plot()
plt.show()
```



Confusion Matrix

```
# Kompilasi model
base_learning_rate = 0.00001
inception_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

Hyperparameter

```
Classification Report:
              precision    recall  f1-score       support
Male           0.961569  0.949407  0.955449   1265.000000
Female         0.947840  0.960363  0.954061   1211.000000
accuracy       0.954766  0.954766  0.954766      0.954766
macro avg      0.954705  0.954885  0.954755   2476.000000
weighted avg   0.954854  0.954766  0.954770   2476.000000
```

Teks paragraf Anda

ARSITEKTUR

VGG

VGG (Visual Geometri Group) adalah arsitektur convolutiinal neural network yang pertama kali diperkenalkan pada tahun 2014 oleh tim peneliti dari Universitas Oxford. Model arsitektur VGG telah menjadi salah satu model pembelajaran yang paling populer dan banyak digunakan untuk visi komputer, termasuk klasifikasi gambar, deteksi objek, dan segmentasi.

```
[ ] import os
    from tensorflow.keras import layers
    from tensorflow.keras import Model
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import tensorflow as tf
```

```
from keras.optimizers import Adam
model = tf.keras.models.Sequential([
    # 1st conv
  tf.keras.layers.Conv2D(96, (11,11),strides=(4,4), activation='relu', input_shape=(218, 178, 3)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2,2)),
    # 2nd conv
  tf.keras.layers.Conv2D(256, (11,11),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 3rd conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 4th conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 5th Conv
  tf.keras.layers.Conv2D(256, (3, 3), strides=(1, 1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2, 2)),
  # To Flatten layer
  tf.keras.layers.Flatten(),
  # To FC layer 1
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  #To FC layer 2
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(2, activation='sigmoid')
  ])


model.summary()
```

Model Arsitektur

TERIMA KASIH
ATAS PERHATIANNYA

Matur Nuhun

For Cooperation and
Collaboration Offer

April
2023