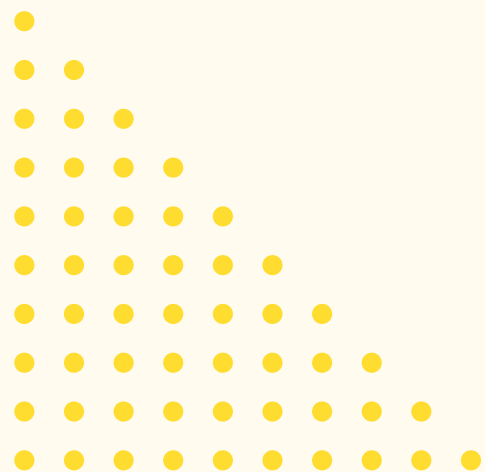
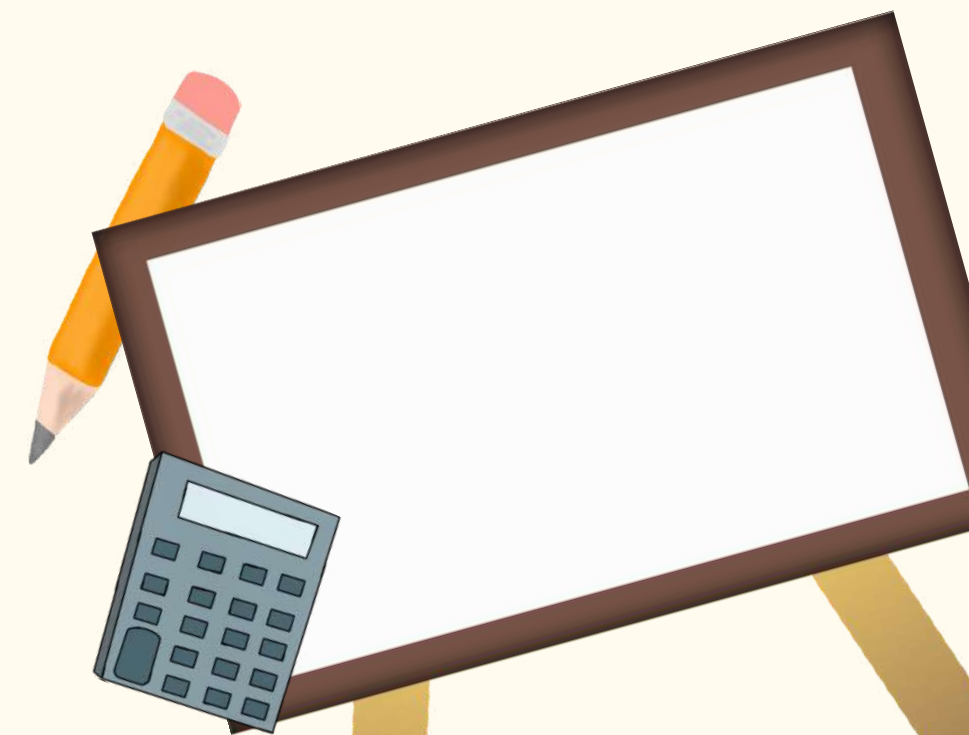


Deteksi Penggunaan Masker



U T S P C D



Kelompok



Reyhan Jarsi Yoga

200810139

Alfian Dorif Murtadlo

200810251

Salma Dian Aprilia

200810151



Pendahuluan



Perkembangan pesat dari transformasi digital saat ini menghasilkan suatu metode pendeteksian objek menggunakan machine learning.

Deteksi objek merupakan salah satu teknik computer vision yang memberikan kemudahan bagi manusia untuk membantu pekerjaan.

Sebuah sistem memerlukan suatu dataset yang akan dibuat untuk mengenai suatu objek. Penelitian sebelumnya meneliti mengenai analisis klasifikasi menggunakan data citra masker dengan cara membandingkan metode CNN dengan kombinasi CNN dan SVM. Pada penelitian tersebut disebutkan jika arsitektur terbaik yang digunakan menggunakan salah satu dari MobileNet, ResNet50, dan VGG-16. Hasil dari penelitian tersebut didapatkan nilai kinerja tertinggi adalah kombinasi dari CNN dengan ResNet50 yang berada pada tingkat akurasi 99,41%.

Tujuan Penelitian



Untuk mengetahui hasil akurasi mengenai klasifikasi dan pendeteksian masker menggunakan metode CNN dengan arsitektur VGG16Net

Untuk mengukur kinerja algoritma dari nilai kinerja precision dan recall dalam studi kasus pendeteksian penggunaan masker



Rumusan Masalah

PERTAMA


Bagaimana cara mengidentifikasi citra bermasker dengan citra tidak bermasker dengan menggunakan metode CNN?

KEDUA

Bagaimana pengaruh ukuran dataset terhadap kinerja penelitian mengenai studi kasus pendeteksian penggunaan masker?

KETIGA

Berapa nilai akurasi tertinggi yang dihasilkan oleh penelitian pengdeteksian penggunaan masker ini?



Data Set

Dataset diperoleh dari internet, tepatnya kaggle yang berjumlah 3.275 menggunakan masker, 3.828 tidak menggunakan masker.

Object detection

Dalam implementasi kelompok kami, kami memanfaatkan fungsi `from mtcnn import MTCNN` untuk melakukan deteksi objek. Teknik ini memungkinkan kami untuk secara efektif mengidentifikasi dan menemukan objek-objek pada gambar atau video.

Secara keseluruhan, deteksi objek merupakan komponen integral dalam pengenalan visual yang memungkinkan sistem untuk memahami konten visual dalam gambar atau video.

Deep Learning dan Machine Learning

Machine learning mencakup algoritma yang otomatis memodelkan prediksi dengan mendeteksi pola dalam data. Deep learning, seperti Convolutional Neural Network (CNN), fokus pada pembentukan model dari data besar dan kompleks, terutama dalam konteks visual seperti gambar.

Dalam studi kasus kami yaitu deteksi penggunaan masker, algoritma CNN digunakan untuk mengenali pola terkait masker pada wajah, memungkinkan klasifikasi otomatis apakah masker digunakan atau tidak. Teknologi ini menjadi kunci dalam mencapai tujuan deteksi penggunaan masker secara otomatis.

Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan gabungan pembelajaran mesin dan jaringan syaraf tiruan yang sangat relevan dalam klasifikasi dan deteksi gambar. Metode CNN memungkinkan otomatisasi proses klasifikasi, ekstraksi objek, dan pengenalan objek pada gambar.

Dalam konteks studi kasus kami, fokusnya adalah pada deteksi penggunaan masker. CNN memainkan peran sentral dalam menerapkan metode efektif untuk mengidentifikasi apakah masker digunakan atau tidak dengan memisahkan pola wajah dengan masker dan tanpa masker. Teknologi ini memfasilitasi tujuan kami untuk mencapai deteksi otomatis penggunaan masker.

Arsitektur VGG16Net

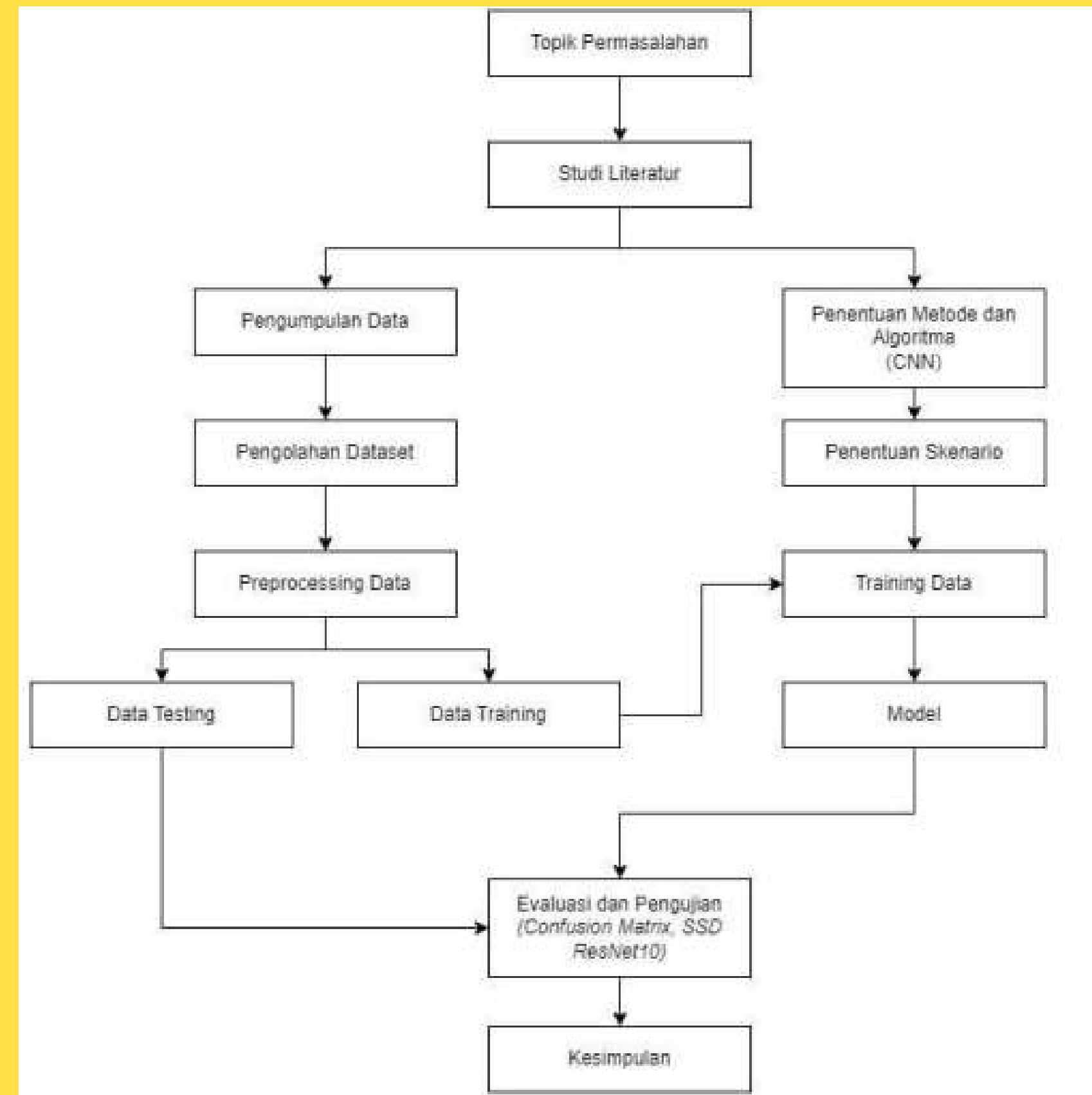
Arsitektur VGG16Net, dikembangkan oleh AlexNet, menonjol dalam pemrosesan ekstraksi fitur pada lapisan konvolusi, memecah gambar menjadi elemen-elemen yang dapat diklasifikasikan. Dengan total 16 lapisan, terdiri dari 13 lapisan konvolusi, 2 lapisan fully connected, dan 1 lapisan untuk klasifikasi.

Dalam implementasi dengan studikamus kami penerapannya digunakan untuk mengolah gambar wajah untuk deteksi penggunaan masker. Arsitektur ini memungkinkan kami untuk membangun model yang efektif dalam mengidentifikasi apakah seseorang menggunakan masker atau tidak melalui proses klasifikasi yang kuat.

Alur Penelitian

Urutan alur secara singkat

1. Penentuan Topik Permasalahan (Klasifikasi Deteksi Penggunaan Masker)
2. Studi Literatur (Tinjauan terhadap Algoritma CNN dan Arsitektur VGG16Net)
3. Penentuan Metode dan Algoritma (Pemilihan Metode Convolutional Neural Network (CNN))
4. Pengumpulan Data (Kumpulan Data Citra Wajah Dengan dan Tanpa Masker)
5. Preprocessing Data (Normalisasi dan Pemrosesan Data Citra)
6. Pengelompokan Data (Pembagian Data untuk Training dan Testing)
7. Penentuan Skema (Arsitektur VGG16Net)
8. Pelatihan Model (Training Data)
9. Evaluasi dan Pengujian Model (Mengukur Akurasi dan Performa Model)
10. Penyajian Hasil (Confusion Matrix, dll.)



Hasil & Pembahasan

- Dataset Mask Detection
- Pembuatan Model
- Pelatihan Model
- Confusion Matrix
- Tahap Pengujian



Dataset Mask Detection

Total Data Set Training 7.553 file
yang terdiri dari :

3.725 Wajah Bermasker

3.828 Wajah Tanpa Masker

Data testing yang digunakan
sebanyak 16 data, 8 data bermasker
dan 8 data tidak bermasker

Link Kaggle : www.kaggle.com/omkargurav/face-mask-dataset



Pembuatan Model

```
# Membentuk bagian head dari model yang akan ditempelkan
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# Menempatkan head model pada base model
model = Model(inputs=baseModel.input, outputs=headModel)

# Perulangan pada seluruh base model
for layer in baseModel.layers:
    layer.trainable = False

# Persiapan kompilasi model
print("Mengkompilasi model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])

model.summary()
```

Tahapan Pembuatan Model :

- Membentuk Head Model
- Menempatkan Head Model yang telah dibuat pada base model
- Looping pada seluruh layer base model

Pelatihan Model

```
# Pelatihan model
print("Training head model...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)
```

Seluruh dataset akan dilakukan training model dengan metode CNN dan arsitektur VGG16Net. Arsitektur dari pelatihan model dilatih selama 30 epoch dan batch size hingga 95.

Confusion Matrix untuk Evaluasi Kinerja Model

```
classes = [0, 1]
# Plot confusion matrix
plt.imshow(conf, interpolation='nearest', cmap=plt.cm.Greens)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

fmt = 'd'
thresh = conf.max() / 2.
for i, j in itertools.product(range(conf.shape[0]), range(conf.shape[1])):
    plt.text(j, i, format(conf[i, j], fmt),
             horizontalalignment="center",
             color="white" if conf[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Prediction Label')
```

Confusion matrix adalah alat pengukuran kinerja proses pelatihan model, terutama pada klasifikasi gambar. Secara umum, confusion matrix adalah Digunakan untuk membandingkan nilai aktual dan nilai prediksi. Pengukuran kinerja mencakup akurasi, presisi, recall, dan F1 score.

Pada Gambar disamping menunjukkan source code untuk menampilkan prediksi label evaluasi menggunakan confusion matrix.

Rumus dan Penghitungan Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (4)$$

Secara garis besar confusion matrix adalah tools analitik untuk dapat memprediksi dengan memberikan perbandingan nilai aktual dan nilai dari prediksi hal tersebut dilakukan sebagai evaluasi kinerja untuk pelatihan model.

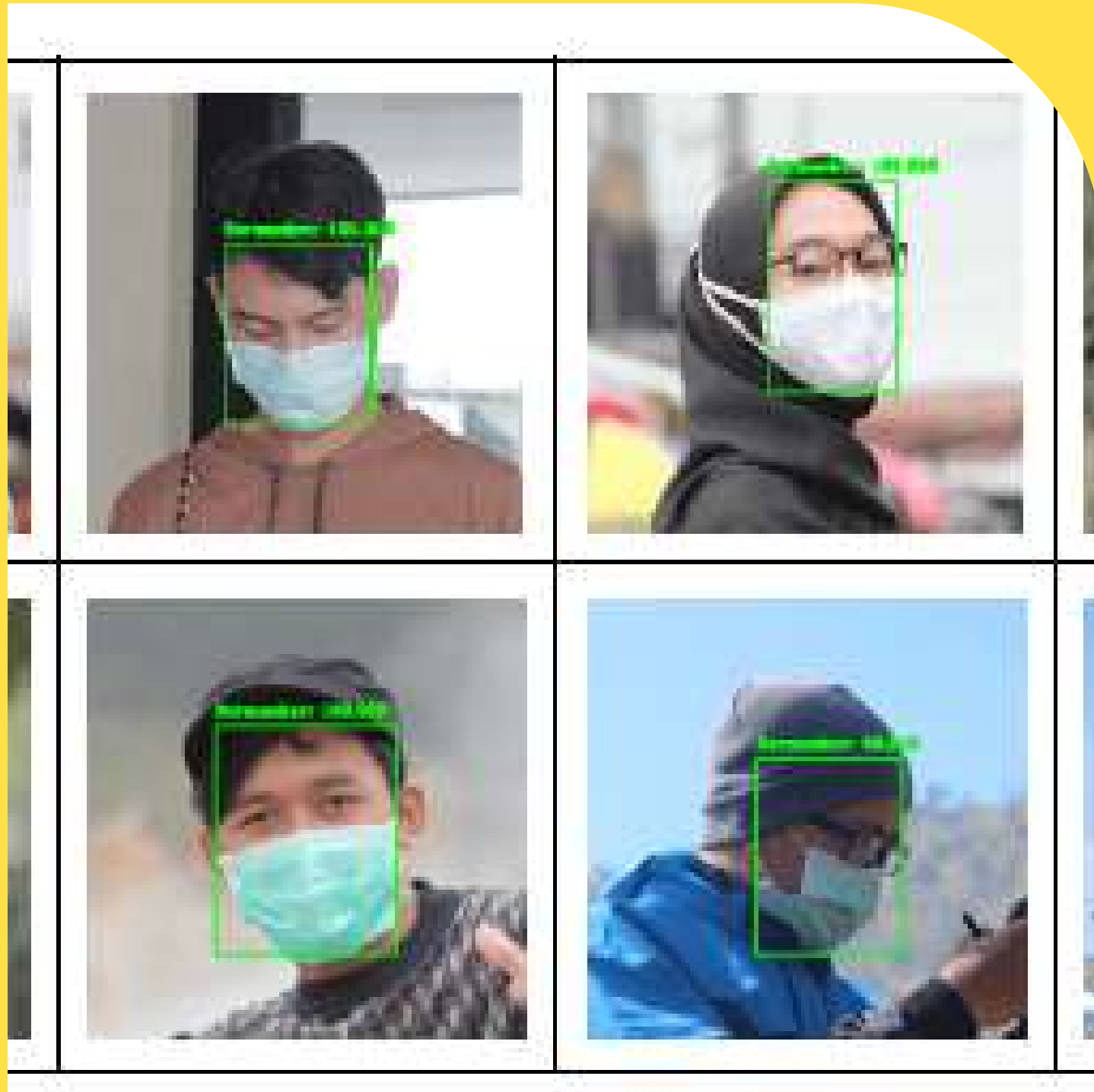
Pada gambar disamping merupakan Rumus perhitungan confusion matrix

Rumus dan Penghitungan Confusion Matrix

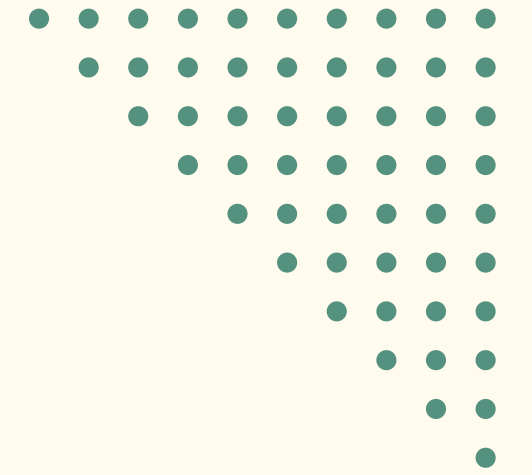
```
Akurasi = 0.992  
Recall = 0.984  
Presisi = 1.000  
F1 = 0.992  
time: 21.7 ms (started: 2022-01-02 10:26:14 +00:00)
```

Penelitian sebelumnya menggunakan CNN dan transfer learning untuk mendeteksi citra wajah bermasker. Arsitektur VGG16 memiliki akurasi sebesar 92,7%, sementara arsitektur Xception mencapai akurasi terbaik yaitu 0,988. Penelitian berikutnya menggunakan confusion matrix untuk mengukur kinerja model, dan hasilnya menunjukkan peningkatan akurasi menjadi 0,992. Hasil evaluasi ini membuktikan peningkatan akurasi dibandingkan penelitian sebelumnya dengan arsitektur VGG16 (92,7%) dan arsitektur Xception (0,988).

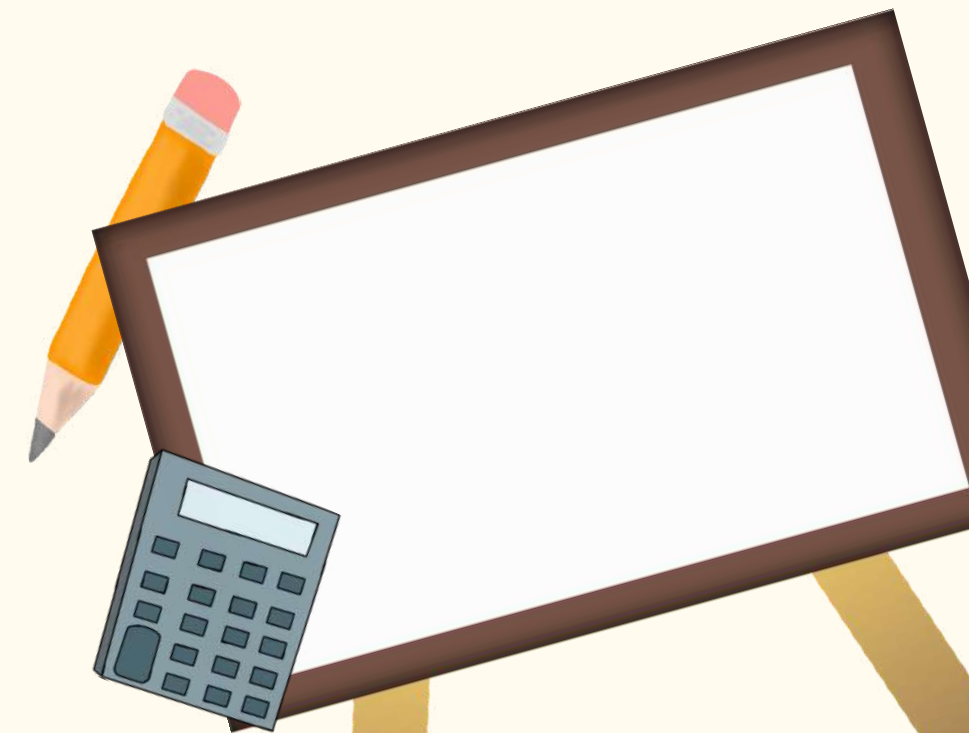
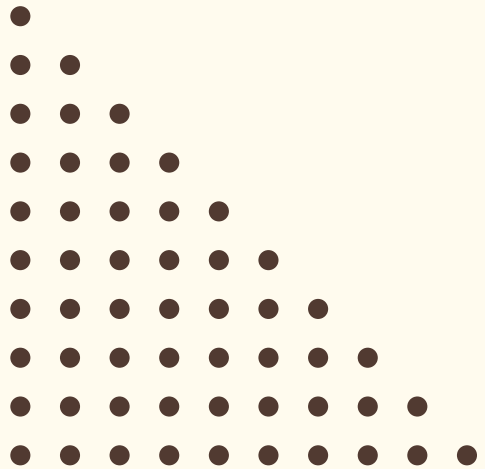
Tahap Pengujian



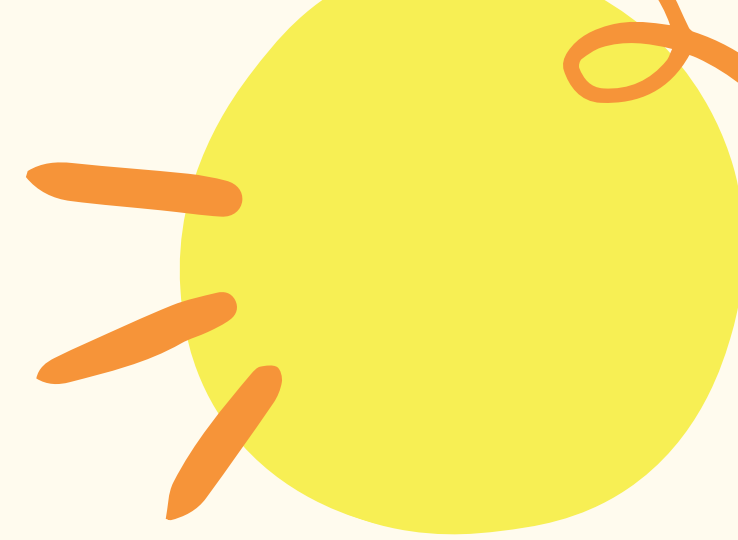
Pada tahap pengujian digunakan data testing, yaitu sampel citra bermasker dan tidak bermasker total berjumlah 16 data testing. Tahap dari pengujian ini, yaitu memasukkan satu per satu sampel dataset citra bermasker dan tidak bermasker



PPT END



Akurasi dalam Code



```
2023-10-17 20:13:57.795006: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/5
 37/189 [====>.....] - ETA: 5:46 - loss: 0.3380 - accuracy: 0.8421C:\Users\fiand\AppData\Local\Programs\Python\Python311\Lib\site-packages\PTL\Image.py:970: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images
  warnings.warn(
189/189 [=====] - 467s 2s/step - loss: 0.1607 - accuracy: 0.9346
Epoch 2/5
189/189 [=====] - 479s 3s/step - loss: 0.0760 - accuracy: 0.9705
Epoch 3/5
189/189 [=====] - 508s 3s/step - loss: 0.0468 - accuracy: 0.9839
Epoch 4/5
189/189 [=====] - 511s 3s/step - loss: 0.0384 - accuracy: 0.9879
Epoch 5/5
189/189 [=====] - 522s 3s/step - loss: 0.0427 - accuracy: 0.9836
C:\Users\fiand\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
Found 1512 images belonging to 2 classes.
1512/1512 [=====] - 165s 109ms/step - loss: 0.0330 - accuracy: 0.9854
Akurasi: 98.5449731349945 %
PS C:\Local Disk D\Kuliah\PCD\PCD_Masker\Masker> |
```