Techical Report

KLASIFIKASI GAMBAR ANEMIA MENGGUNAKAN K-NEAREST NEIGHBORS (KNN)

Dibuat oleh:

Ariel Yosua Hasibuan (105222004)

Haekal Putra Alharis (105222028)

Bintang Akbar Alim (105222037)

Abstrak

Proyek ini bertujuan untuk melakukan klasifikasi gambar guna mendeteksi kondisi anemia menggunakan algoritma K-Nearest Neighbors (KNN). Dataset terdiri dari dua kelas gambar, yaitu "anemic" dan "nonanemic", yang masing-masing dikumpulkan dalam folder terpisah. Proses utama yang dilakukan mencakup eksplorasi data (EDA), preprocessing, pelatihan model KNN, dan evaluasi model menggunakan akurasi dan confusion matrix. Model KNN menunjukkan performa klasifikasi yang cukup baik dengan hasil akurasi yang signifikan pada data uji.

Pendahuluan

Deteksi dini anemia sangat penting untuk menghindari komplikasi kesehatan yang serius. Salah satu pendekatan untuk mengidentifikasi anemia secara otomatis adalah dengan menggunakan teknik machine learning berbasis citra. Dalam proyek ini, digunakan pendekatan supervised learning dengan algoritma KNN untuk mengklasifikasikan gambar ke dalam dua kelas: anemic dan nonanemic.

Metodologi

a. Dataset

   Dataset yang digunakan terdiri dari 2 folder, yaitu:

   1. anemic, folder berisi gambar kondisi anemia.
   2. nonanemic, folder berisi gambar kondisi non-anemia.

   Distribusi data:

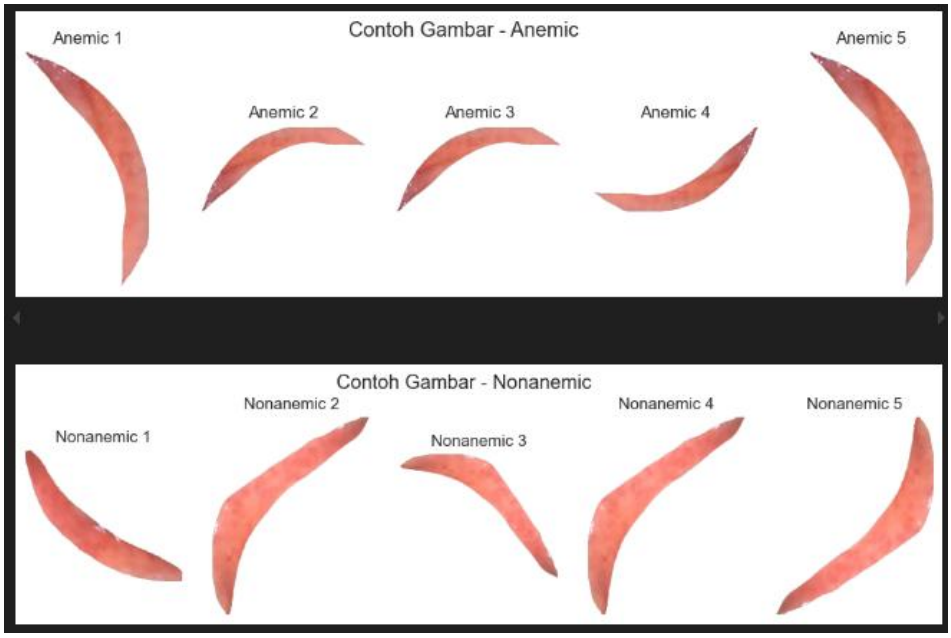
   1. Jumlah gambar pada kelas anemic yaitu : 2563
   2. Jumlah gambar pada kelas nonanemic yaitu = 1714

b. Exploratory Data Analysis

Dilakukan visualisasi lima gambar contoh dari masing-masing kelas untuk memahami karakteristik visual dasar dari citra. Selain itu, ditampilkan perbandingan jumlah gambar antara dua kelas menggunakan diagram batang.

```python
def show_sample_images(folder, title):
    plt.figure(figsize=(12, 3))
    for idx, filename in enumerate(os.listdir(folder)[:5]):
        img_path = os.path.join(folder, filename)
        img = Image.open(img_path)
        plt.subplot(1, 5, idx+1)
        plt.imshow(img)
        plt.title(f"{title} {idx+1}")
        plt.axis('off')
    plt.suptitle(f"Contoh Gambar - {title}")
    plt.show()

show_sample_images(anemic_path, "Anemic")
show_sample_images(nonanemic_path, "Nonanemic")
```

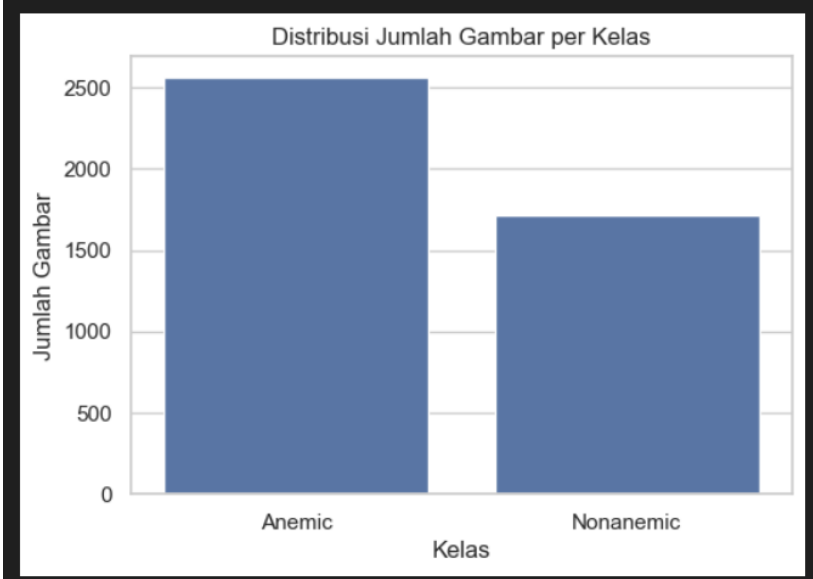


```python
df_summary = pd.DataFrame({
    'Kelas': ['Anemic', 'Nonanemic'],
    'Jumlah Gambar': [n_anemic, n_nonanemic],
    'Persentase (%)': [n_anemic / (n_anemic + n_nonanemic) * 100,
                       n_nonanemic / (n_anemic + n_nonanemic) * 100]
})
print("Ringkasan Dataset:")
display(df_summary)

plt.figure(figsize=(6, 4))
sns.barplot(x='Kelas', y='Jumlah Gambar', data=df_summary)
plt.title("Distribusi Jumlah Gambar per Kelas")
plt.ylabel("Jumlah Gambar")
plt.xlabel("Kelas")
plt.show()
```

Ringkasan Dataset:

	Kelas	Jumlah Gambar	Persentase (%)
0	Anemic	2563	59.925181
1	Nonanemic	1714	40.074819



c. Preprocessing

Setiap gambar diproses sebagai berikut:

- Resize ke ukuran tetap (64x64 piksel).
- Konversi ke grayscale untuk menyederhanakan fitur.
- Normalisasi piksel ke rentang [0, 1].
- Flatten citra menjadi vektor 1 dimensi untuk input model.

```python
IMG_SIZE = (64, 64)

def load_images_from_folder(folder, label):
    images = []
    labels = []
    for filename in os.listdir(folder):
        if filename.lower().endswith((".jpg", ".jpeg", ".png")):
            img_path = os.path.join(folder, filename)
            img = Image.open(img_path).resize(IMG_SIZE).convert('L')
            img_array = np.array(img).flatten() / 255.0
            images.append(img_array)
            labels.append(label)
    return images, labels

anemic_images, anemic_labels = load_images_from_folder(anemic_path, 1)
nonanemic_images, nonanemic_labels = load_images_from_folder(nonanemic_path, 0)

X = np.array(anemic_images + nonanemic_images)
y = np.array(anemic_labels + nonanemic_labels)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
```

d. Pemodelan dan Pelatihan

- Algoritma: K-Nearest Neighbors (KNN).
- Parameter: k = 3
- Data dibagi menggunakan train_test_split (rasio 80:20) dengan stratify agar distribusi label tetap proporsional.
- Model KNN dilatih menggunakan data training.

```python
IMG_SIZE = (64, 64)

def load_images_from_folder(folder, label):
    images = []
    labels = []
    for filename in os.listdir(folder):
        if filename.lower().endswith((".jpg", ".jpeg", ".png")):
            img_path = os.path.join(folder, filename)
            img = Image.open(img_path).resize(IMG_SIZE).convert('L')
            img_array = np.array(img).flatten() / 255.0
            images.append(img_array)
            labels.append(label)
    return images, labels

anemic_images, anemic_labels = load_images_from_folder(anemic_path, 1)
nonanemic_images, nonanemic_labels = load_images_from_folder(nonanemic_path, 0)

X = np.array(anemic_images + nonanemic_images)
y = np.array(anemic_labels + nonanemic_labels)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Evaluasi Model

a. Akurasi

Model KNN mendapat akurasi senilai:

```
y_pred = knn.predict(X_test)

print(f" Akurasi Model KNN: {accuracy_score(y_test, y_pred):.2f}")
```

```
Akurasi Model KNN: 0.75
```

b. Classification Report

```
print("\n Classification Report:")
print(classification_report(y_test, y_pred, target_names=["Nonanemic", "Anemic"]))
```

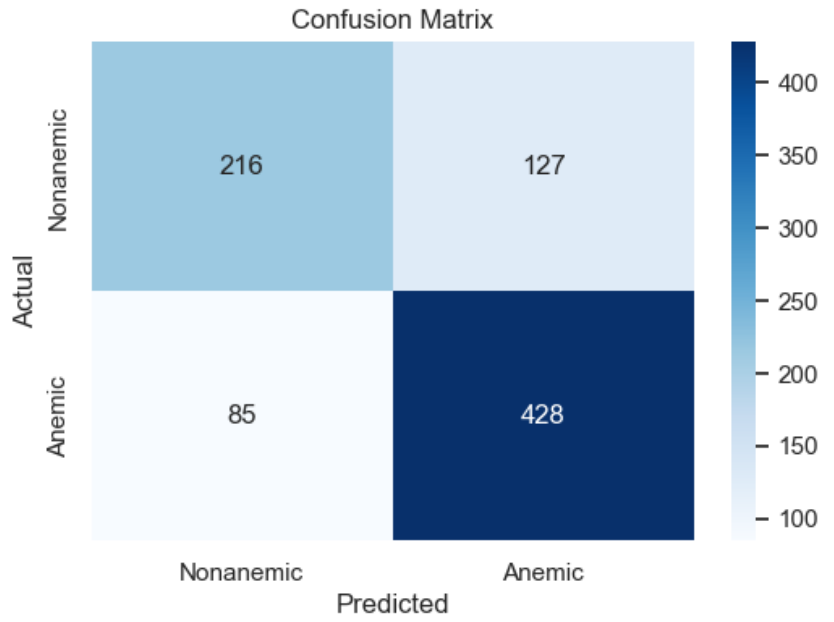
```
Classification Report:
              precision    recall  f1-score   support

   Nonanemic       0.72      0.63      0.67       343
      Anemic       0.77      0.83      0.80       513

    accuracy                           0.75       856
   macro avg       0.74      0.73      0.74       856
weighted avg       0.75      0.75      0.75       856
```

c. Confusion Matrix

```python
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Nonanemic', 'Anemic'],
            yticklabels=['Nonanemic', 'Anemic'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



d. Precision & Recall

Untuk kelas Anemic (positif):

- Precision = TP / (TP + FP) = 428 / (428 + 127) ≈ 77.1%
- Recall = TP / (TP + FN) = 428 / (428 + 85) ≈ 83.4%
- F1-score ≈ 80.1%

Untuk kelas Nonanemic (negatif):

- Precision = 216 / (216 + 85) ≈ 71.8%
- Recall = 216 / (216 + 127) ≈ 62.9%
- F1-score ≈ 67.0%

Kesimpulan

Pada proyek ini, telah dilakukan klasifikasi citra menjadi dua kelas yaitu anemic dan nonanemic menggunakan algoritma K-Nearest Neighbor (KNN). Dataset terdiri dari gambar grayscale yang telah diresize, dinormalisasi, dan di-flatten. Model dilatih dengan nilai k=3 dan data dibagi menjadi data latih dan data uji dengan rasio 80:20 secara stratified.

Hasil evaluasi menunjukkan bahwa model KNN memberikan akurasi sebesar 75%. Berdasarkan confusion matrix, model dapat mengklasifikasikan gambar anemic dengan lebih baik dibandingkan gambar nonanemic, ditandai dengan nilai recall untuk kelas anemic yang cukup tinggi (83.4%).

Model juga berhasil mencapai:

- Precision (anemic): 77.1%
- Recall (anemic): 83.4%
- F1-Score (anemic): 80.1%

Saran

1. Tuning Parameter KNN
    - Coba berbagai nilai k (misalnya 1–15) untuk mencari performa terbaik menggunakan teknik seperti cross-validation.
2. Mengurangi Dimensi Fitur
    - Karena flattening gambar menghasilkan fitur berdimensi tinggi (64×64 = 4096), bisa dicoba dimensionality reduction seperti PCA untuk mempercepat training dan menghindari overfitting.
3. Augmentasi Data
    - Lakukan data augmentation (rotasi, flip, zoom) untuk menambah variasi citra, terutama jika terdapat ketidakseimbangan jumlah antar kelas.
4. Bandingkan dengan Model Lain
    - Untuk peningkatan performa, bisa dilakukan perbandingan model dengan SVM atau Random Forest, atau bahkan melangkah ke model deep learning seperti CNN yang lebih kuat untuk klasifikasi gambar.