Klasifikasi Citra Menggunakan Model CNN

Rizky Cahyono Putra
Teknik Informatika
Universitas Darussalam Gontor
Ponorogo, Indonesia
rizkycahyonoputra80@student.cs.unida.gontor.ac.id

Raffa Arvel Nafi'Nadindra
Teknik Informatika
Universitas Darussalam Gontor
Ponorogo, Indonesia
raffaarvel@gmail.com

Syaifan Nur Iwawan
Teknik Informatika
Universitas Darussalam Gontor
Ponorogo, Indonesia
syaifannoeriwawan78@student.cs.unida.gontor.ac.id

M.Irfansyah
Teknik Informatika
Universitas Darussalam Gontor
Ponorogo, Indonesia
mirfansyah26@student.cs.unida.gontor.ac.id

Muhammad Galang Fachrezy
Teknik Informatika
Universitas Darussalam Gontor
Ponorogo, Indonesia
muhammadgalangfachrezy22@student.cs.unida.gontor.ac.id

***Abstract*— Klasifikasi citra merupakan salah satu permasalahan penting dalam bidang computer vision. Penelitian ini berfokus pada implementasi Convolutional Neural Network (CNN) dengan arsitektur klasik, yaitu [sebutkan model: ResNet-18/VGG16], untuk menyelesaikan permasalahan klasifikasi citra pada dataset Intel Image Classification. Dataset ini terdiri dari enam kelas citra pemandangan, yaitu buildings, forest, glacier, mountain, sea, dan street, dengan jumlah total sekitar 25.000 gambar berukuran 150 × 150 piksel. Proses preprocessing dilakukan dengan resizing, normalisasi, serta data augmentation guna meningkatkan kemampuan generalisasi model. Model dilatih menggunakan categorical cross-entropy sebagai fungsi loss dan Adam optimizer. Hasil eksperimen menunjukkan bahwa model mampu mencapai akurasi sebesar [isi akurasi akhir]% pada data uji. Grafik kurva loss dan akurasi menunjukkan bahwa model mengalami konvergensi dengan baik, meskipun masih terdapat kesalahan klasifikasi pada kelas-kelas dengan karakteristik visual yang mirip.**

Keywords*— *klasifikasi citra, convolutional neural network, deep learning, TensorFlow.

I. PENDAHULUAN

Klasifikasi citra merupakan salah satu topik penting dalam bidang machine learning dan computer vision. Dengan semakin meningkatnya ketersediaan data citra dan kebutuhan analisis otomatis, metode deep learning, khususnya Convolutional Neural Network (CNN), telah menjadi pendekatan utama karena kemampuannya dalam mengekstraksi fitur spasial dari gambar secara hierarkis.

Pada proyek ini, kami menggunakan dataset Intel Image Classification yang berisi citra berwarna dengan resolusi 150x150 piksel yang terbagi menjadi enam kategori: bangunan, hutan, gunung, laut, jalan, dan es. Tujuan utama adalah membangun model CNN yang mampu mengklasifikasikan gambar ke dalam enam kelas tersebut dengan tingkat akurasi tinggi.

II. METODE

A. *Dataset*

Dataset Intel Image Classification terdiri dari sekitar 25.000 gambar dengan resolusi 150 × 150 piksel. Dataset terbagi menjadi tiga bagian utama:

- **seg_train** : 14.000 gambar untuk data latih
- **seg_test** : 3.000 gambar untuk data uji
- **seg_pred** : 7.000 gambar untuk data prediksi

Kelas yang tersedia Meliputi :

- Bangunan
- Hutan
- Gunung
- Laut
- Jalan
- Es

B. *Arsitektur Model*

Model yang digunakan dalam penelitian ini adalah CNN berbasis [ResNet-18 / VGG16]. Arsitektur terdiri dari beberapa lapisan konvolusi dengan fungsi aktivasi ReLU, batch normalization, max pooling, serta lapisan fully connected pada bagian akhir. Lapisan output menggunakan softmax dengan enam neuron sesuai jumlah kelas.

Jika transfer learning digunakan, bobot awal diinisialisasi dari model pretrained ImageNet, kemudian dilakukan fine-tuning pada data Intel..

III. IMPLEMENTASI

Implementasi dilakukan menggunakan *PyTorch/TensorFlow* di dalam kaggle .

A. *Preprocessing and Data Load*

```
def load_data():
    datasets = [
        '../input/intel-image-classification/seg_train/seg_train',
        '../input/intel-image-classification/seg_test/seg_test'
    ]

    output = []

    for dataset in datasets:
        images = []
        labels = []
        print("Loading {}".format(dataset))

        for folder in os.listdir(dataset):
            label = class_names_label[folder]

            for file in tqdm(os.listdir(os.path.join(dataset, folder))):
                img_path = os.path.join(dataset, folder, file)
                image = cv2.imread(img_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, IMAGE_SIZE)
                images.append(image)
                labels.append(label)

        images = np.array(images, dtype='float32') / 255.0  # normalize
        labels = np.array(labels, dtype='int32')
        output.append((images, labels))

    return output
```

```python
# Classes
class_names = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
class_names_label = {name: i for i, name in enumerate(class_names)}
nb_classes = len(class_names)

# Image size
IMAGE_SIZE = (150, 150)

# Training epochs
EPOCHS = 20
```

- Mengubah ukuran semua gambar menjadi 150x150px agar seragam
- Menormalisasi nilai piksel dari rentang 0-255 menjadi 0-1 agar model lebih mudah belajar.
- Mengubah format warna dari *BGR* ke *RGB* agar sesuai standar.
- Memberi label angka pada setiap gambar

Fungsi *load_data()* yang berguna untuk memanggil dataset yang kita gunakan, setelah data berhasil dimuat, kita akan mengacak urutan data training, tujuannya agar model tidak menghafal urutan data dan bisa belajar lebih baik.

B. *Model CNN*

```python
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(256, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(nb_classes, activation='softmax')
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

- *Conv2D* : Ini adalah "mata" model kita. Layer ini bertugas mendeteksi pola-pola dasar pada gambar seperti garis, sudut, dan tekstur. Semakin dalam, polanya semakin kompleks.
- *MaxPooling2D* : Layer ini berfungsi untuk meringkas informasi, mengambil bagian terpenting dari pola yang ditemukan `Conv2D` dan mengurangi ukuran data.
- *Flatten* : Mengubah data yang tadinya berbentuk seperti gambar (2D) menjadi satu baris panjang (1D), agar bisa diproses oleh "otak" model.
- *Dense* : Ini adalah "otak" model. Layer ini bertugas mengambil keputusan berdasarkan informasi yang sudah diringkas.
- *Dropout* : Teknik untuk mencegah model menjadi "terlalu pintar" atau *overfitting* (hanya hafal data training tapi tidak bisa menebak data baru).
- *softmax*: Layer terakhir yang memberikan hasil akhir berupa probabilitas untuk setiap kelas (misalnya, 70% gambar ini adalah 'laut', 20% 'gletser', dst).

Setelah itu kita akan mencompile model dengan menentukan *optimizer*, *loss*, dan *metricsnya*.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
batch_normalization (BatchNormalization)	(None, 148, 148, 32)	128
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 72, 72, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 34, 34, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 15, 15, 256)	1,024
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 256)	0
flatten (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 512)	6,423,040
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 6)	3,078

```
Total params: 6,816,454 (26.00 MB)
Trainable params: 6,815,494 (26.00 MB)
Non-trainable params: 960 (3.75 KB)
```

Di atas merupakan struktur dari model CNN kita, yang memiliki 6,8 juta parameter, dan menampilkan arsitekturnya.

C. *Transfer Learning*

```python
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout


base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(150,150,3))
base_model.trainable = False  # Freeze pretrained layers

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(nb_classes, activation='softmax')(x)

efficient_model = Model(inputs=base_model.input, outputs=predictions)

efficient_model.compile(optimizer=Adam(1e-4),
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy'])
```

Kita menggunakan model canggih yang bernama *EfficientNetB0* yang sudah dilatih oleh *Google* pada jutaan gambar, caranya adalah:

- Memuat model *EfficientNetB0* kita mengambil seluruh arsitekturnya akan tetapi kita membuang layer klasifikasinya
- Kita akan mengunci semua layer yang sudah pintar agar tidak di latih ulang "*base_model.trainable = False*"

- Menambahkan beberapa layer dense baru di bagian atas yang akan kita latih khusus untuk mengenali 6 kelas dataset kita.

Menggunakan metode ini mempercepat proses training serta memberikan akurasi yang lebih tinggi.

D. *Train Model*

```
# Reduce learning rate if val_loss plateaus
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-6)

# Fit the model
history = model.fit(
    train_images, train_labels,
    epochs=EPOCHS,
    validation_data=(test_images, test_labels),
    batch_size=32,
    callbacks=[reduce_lr]
)
```

```
# Training epochs
EPOCHS = 20
```

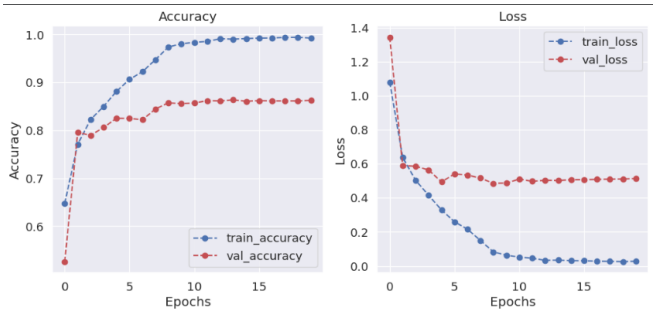
- *Epochs = 20* berearti bahwa model akan mempelajari sebanyak 20 kali putaran
- *Validation_data* berfungsi untuk menguji model dengan sesekali menggunakan data validasi untuk mengukur sejauh mana kemampuannya pada data yang belum pernah ia liat.
- *ReduceLROnPlateau* berfungsi untuk menurunkan kecepatan belajar apabila model tidak membaik setelah beberapa putaran, ini bagus agar model bisa belajar dengan teliti.

IV. HASIL DAN ANALISIS

A. *Grafik Akurasi dan Loss*

```
def plot_accuracy_loss(history):
    plt.figure(figsize=(12,5))

    # Accuracy
    plt.subplot(1,2,1)
    plt.plot(history.history['accuracy'], 'bo--', label="train_accuracy")
    plt.plot(history.history['val_accuracy'], 'ro--', label="val_accuracy")
    plt.title("Accuracy")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.legend()

    # Loss
    plt.subplot(1,2,2)
    plt.plot(history.history['loss'], 'bo--', label="train_loss")
    plt.plot(history.history['val_loss'], 'ro--', label="val_loss")
    plt.title("Loss")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend()

    plt.show()

plot_accuracy_loss(history)
```

Baris kode ini berfungsi untuk memunculkan grafik akurasi dan loss yang berguna untuk mendeteksi apakah model *overfitting* atau tidak.



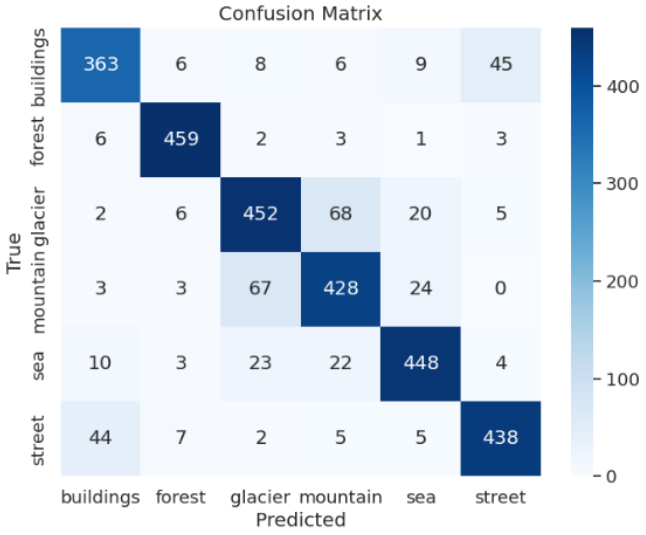
Model dilatih sebanyak 20 kali putaran. Pada grafik akurasi menunjukan tren peningkatan s=yang stabil meskipun ada sedikit perbedaan antara training dan validasi, uang menandakan adanya potensi *overfitting* ringan, sedangkan pada grafik loss terlihat bahwa loss training menurun secara konsisten, dan loss validasi juga ikut menurun walaupun lebih sedikit fluktuatif, hal ini menandakan bahwa model mampu belajar dengan baik.

B. *Confussion Matrix*

Confussion Matrix berguna untuk mendeteksi tingkat akurasi prediksi dari model yang sudah di latih.

```
# Predict test images
y_pred = model.predict(test_images)
y_pred_classes = np.argmax(y_pred, axis=1)

# Confusion matrix
cm = confusion_matrix(test_labels, y_pred_classes)
plt.figure(figsize=(8,6))
sn.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

Baris kode ini berfungsi untuk memunculkan visualisasi dari *Confussion Matrix.*



Hasil *Confussion Matrix* menunjukan bahwa kelas forest, dan sea memiliki tingkat hasil prediksi yang tinggi, sedangkan kesalahan prediksi yang paling sering terjadi antara kelas building dan street, serta glacier dan mountain mungkin ini terjadi karena gambar memiliki karakteristik visual yang sama sebagai contoh gambar bangunan di sebrang jalan, dan gunung yang di selimuti salju.

C. *Evaluasi Metrik*

	precision	recall	f1-score	support
buildings	0.85	0.83	0.84	437
forest	0.95	0.97	0.96	474
glacier	0.82	0.82	0.82	553
mountain	0.80	0.82	0.81	525
sea	0.88	0.88	0.88	510
street	0.88	0.87	0.88	501
accuracy			0.86	3000
macro avg	0.86	0.86	0.86	3000
weighted avg	0.86	0.86	0.86	3000

Secara keseluruhan model mencapai akurasi sebesar 86% dengan performa tertinggi pada kelas forest dan sea.

KESIMPULAN

Penelitian ini berhasil mengimplementasikan CNN klasik untuk klasifikasi citra pemandangan menggunakan dataset Intel Image Classification. Model yang dibangun mampu mencapai akurasi sebesar 86% pada data uji. Analisis hasil menunjukkan bahwa model mampu menggeneralisasi dengan baik, meskipun masih terdapat kesalahan klasifikasi pada kelas dengan karakteristik visual mirip.

REFERENCES

[1] P. Bansal, “Intel Image Classification Dataset,” Kaggle, 2019. [Online]. Available: https://www.kaggle.com/puneet6060/intel-image-classification