BAB IV

HASIL DAN PEMBASAN

Pada bab ini akan membahas perancangan dan melakukan klasifikasi gambar kue tradisional menggunakan *Convolutional Neural Network (CNN)*. Perancangan sistem klasifikasi dilakukan dengan melatih gambar kue dengan algoritma *CNN* guna menghasilkan sebuah model yang bagus dengan akurasi terbaik. Dalam pelatihan *CNN* terdapat data train dan data test serta data validasi. Data train digunakan untuk melakukan training pada model, data validasi digunakan untuk memvalidasi data train saat pelatihan model dan data test digunakan validasi dari data training.

4.1. Analisis Dataset

Dataset dipersiapkan untuk dipelajari oleh model, set data yang telah diunduh kemudian akan dilakukan tahap *preprocessing*. Dalam *preprocessing* dataset tersebut, set data akan dipelajari dari segi tampilan gambarnya, ukuran gambarnya, ukuran *shape* gambar dan proporsi masing-masing kelas, tujuannya yaitu untuk mendapatkan *insight* terhadap dataset.

Dataset yang telah di unduh dari *https://atapdata.ai*. Dalam dataset tersebut terdapat 3 folder yaitu *train, validation,* dan *test* dan 8 label atau kelas. Dari masing-masing folder berisikan 8 folder jenis kue tradisional dan hanya diambil 6 kue tradisional saja. Dibawah ini pada **tabel 4.1** merupakan rincian data gambar pada masing-masing folder *train, validation,* dan *test*.

Tabel 4.1 Rincian dataset dari atapdata

LABEL	JUMLAH TRAIN	JUMLAH VALIDATION	JUMLAH TEST
Kue Dadar Gulung	192	20	20
Kue Klepon	200	20	20
Kue Lapis	201	20	20

Lanjutan **Tabel 4.1** Rincian dataset dari atapdata

LABEL	JUMLAH TRAIN	JUMLAH VALIDATION	JUMLAH TEST
Kue Lumpur	198	20	20
Kue Risoles	196	20	20
Kue Serabi	181	20	20
JUMLAH	**1168**	**120**	**120**

Pada **Tabel 4.2** dibawah ini merupakan rincian dataset yang diunduh dari penelusuran *google image* yang dikumpulkan secara mandiri.

Tabel 4.2 Rincian dataset dari *google image*

LABEL	JUMLAH TRAIN	JUMLAH VALIDATION	JUMLAH TEST
Kue Cucur	200	20	20
Kue Onde-onde	200	20	20
JUMLAH	**400**	**40**	**40**

Jadi total sampel yang digunakan keseluruhan adalah 1.888 data gambar kue tradisional, 1.568 pada data latih, 160 pada data validasi dan 160 pada data tes dari 8 data gambar kue tradisional.

Gambar **4.1** dibawah merupakan sampel data gambar kue tradisional pada masing-masing kelas, ada 8 jenis kelas kue yaitu Kue Klepon, Kue Lumpur, Kue Cucur, Kue Onde-onde Serabi, Kue Dadar Gulung, Kue Lapis dan Kue Risoles.

Gambar 4.1 Sampel dataset

4.2. Preprocessing

Pada dataset yang telah diunduh memiliki ukuran dimensi yang berbeda-beda. *Preprocessing* atau pra-pemrosesan dilakukan untuk menormalisasi dataset gambar menggunakan *Image Augmentation* pada *Keras* dan merubah semua dimensi gambar ke dalam bentuk yang sama. Dengan augmentasi gambar juga dapat memperbanyak data latih agar data latih lebih bervariasi tanpa perlu mencari data yang baru, yaitu dengan cara menduplikasi data gambar. Berikut beberapa parameter yang digunakan untuk memvariasi dataset:

1. *Rescale*, normalisasi *pixel* gambar antara 1-255.
2. *Rotation Range*, merotasi gambar secara acak.
3. *Shear Range*, peregangan gambar atau pergeseran sudut dari titik tertentu secara acak.
4. *Zoom Range*, memperbesar gambar secara acak.

5. *Height Shift Range*, peninggian gambar secara acak. dengan menentukan batas pecahan dari total tinggi dan digeser ke atas atau ke bawah secara acak.
6. *Width Shift Range*, pergeseran gambar dengan menentukan batas pecahan dari total lebar dan digeser ke kiri atau ke kanan secara acak.
7. *Horizontal flip*, memutar balik gambar secara horizontal.
8. *Vertical flip*, memutar balik gambar secara vertikal.
9. *Fill mode nearest*, pengisian *pixel* yang kosong pada gambar yang telah di augmentasi.
10. *Validation Split*, membagi sebagian data untuk validasi dari data latih.

```
train_datagen = ImageDataGenerator(
    #normalisasi gambar
    rescale= 1./255,
    #rotasi gambar sampai 40 derajad
    rotation_range=40,
    #pangkas gambar sampai 20% dari ukuran asli
    shear_range=0.2,
    #zoom gambar sampai 20% dari ukuran asli
    zoom_range=0.2,
    #peninggian gambar sampai 20%
    height_shift_range=0.2,
    #pelebaran gambar sampai 20% dari ukuran asli
    width_shift_range=0.2,
    #putar balik gambar secara horizontal
    horizontal_flip=True,
    #putar balik gambar secara vertical
    vertical_flip=True,
    #isi piksel kosong dengan nilai terdekat
    fill_mode = 'nearest')

validation_datagen = ImageDataGenerator(
    #normalisasi gambar
    rescale= 1./255,
    # split data validasi
    validation_split=0.2)
```

Gambar 4.2 *Parameter Image Generator*

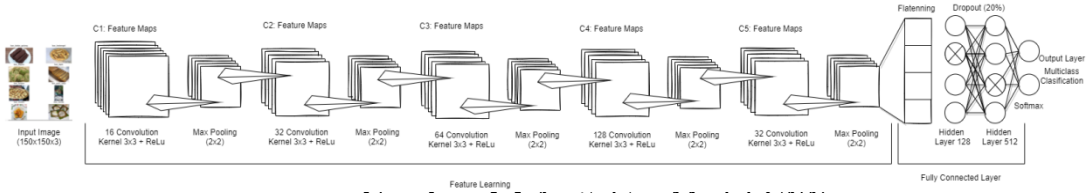
Setelah melakukan konfigurasi parameter pada *Image Augmentation*, tahap selanjutnya adalah menentukan ukuran *batch size*, ukuran target gambar yang akan dihasilkan dan membagi gambar yang akan di generator ke data train dan data validasi. Pada penelitian ini digunakan ukuran gambar sebesar 150x150.

```
#  laporan
BATCH_SIZE = 64


train_generator = train_datagen.flow_from_directory(
    folder_train,
    target_size=(150,150),
    class_mode='categorical',
    shuffle=True,
    batch_size=BATCH_SIZE,
    subset='training')

validation_generator = validation_datagen.flow_from_directory(
    folder_train,
    target_size=(150,150),
    class_mode='categorical',
    shuffle=True,
    batch_size=BATCH_SIZE,
    subset='validation')
```

Gambar 4.3 *Image Augmentation*

4.3. Arsitektur Model *CNN*

Pada model *CNN* yang akan dibuat menggunakan *API Keras* yang berjalan diatas *Tensorflow* (Tensorflow, 2021). Model yang dibuat berbentuk *Sequential*, penelitian ini menggunakan ukuran kernel yaitu 3x3. Ada beberapa susunan layer yang ada di dalam model *Sequential* ini. Pada lapis pertama ada *layer* konvolusi, pada layer konvolusi ini semua nilai pixel gambar berbentuk larik atau vektor dimana fungsinya untuk mengekstraksi atribut gambar. Pada penelitian ini menggunakan 5 konvolusi layer dengan neuron 16, 32, 64, 128 dan 32. Pada lapis kedua ada *max pooling layer*, *max pooling* selalu berada setelah konvolusi layer, *max pooling layer* berfungsi untuk mereduksi resolusi gambar agar proses ketika pelatihan lebih cepat. Setelah lapisan konvolusi, ada lapisan *hidden layer*. Pada layer ini, ada 3 lapisan *hidden layer* yang digunakan dengan fungsi aktivasi yang umum digunakan untuk klasifikasi gambar yaitu fungsi aktivasi relu dengan variasi *perceptron* 128, 512, dan 8. Pada lapis hidden layer yang terakhir dengan perceptron 8 menggunakan fungsi aktivasi *softmax* untuk klasifikasi banyak kelas atau *multiclass*.



Gambar 4.4 Arsitektur Model *CNN*

4.4. Klasifikasi *CNN*

Setelah model dibuat dan data telah dilakukan preprocessing kemudian siap untuk dilatih dan dilakukan pengklasifikasian dengan algoritma *CNN*.

4.4.1. *Training Model*

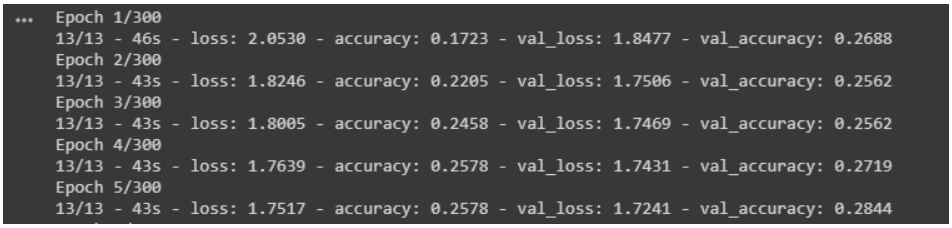
Proses *training* terhadap model dilakukan dengan memanggil fungsi *fit()* dan mengisi parameter pada fungsi tersebut. Parameter tersebut terdiri dari iterasi atau *epoch* dari pelatihan yang nantinya akan ditampilkan untuk melihat pergerakan proses pelatihan.

```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=300,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    verbose=2,
    callbacks=[callbacks]
    )
```

Gambar 4.5 *Training Model*

Seperti yang terlihat pada **Gambar 4.5** pada baris pertama melatih model dengan memanggil fungsi *fit()* dan ditampung pada variabel *history*. Pada baris kedua *train generator* adalah semua data gambar dari data latih. Pada baris ketiga, parameter *step per epoch* adalah jumlah dari data latih dibagi dengan *batch size* atau jumlah file per *epoch*. Pada baris ke empat, parameter *epochs* merupakan banyaknya iterasi pelatihan yang akan dilakukan. Pada baris ke lima, parameter validation_data, diisi dengan data validasi yang akan memvalidasi data latih saat proses pelatihan. Pada baris ke enam *validation steps* adalah jumlah dari data validasi dibagi dengan *batch size* atau jumlah file per *epoch*. Pada baris ke tujuh, *verbose* merupakan parameter untuk melihat proses kemajuan per *epoch* saat pelatihan. Pada baris terakhir, *callbacks* merupakan fungsi untuk memantau kinerja model, pada *callbacks* disini yang di pantau adalah *accuracy*, jika *accuracy* mencapai hasil yang sudah ditentukan maka

proses pelatihan akan dihentikan secara otomatis sebelum mencapai batas *epoch*.



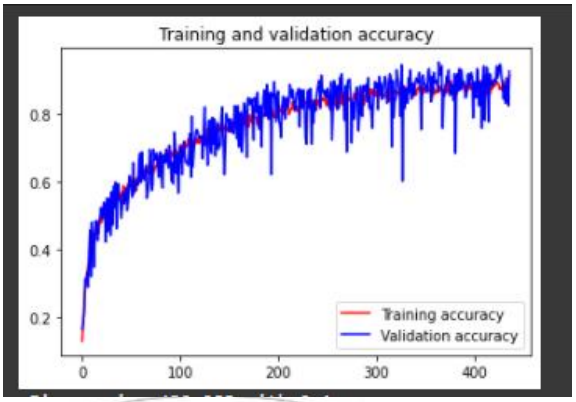
Gambar 4.6 Proses pelatihan tiap *epoch*

Terlihat pada **Gambar 4.6** bahwa iterasi atau *epoch* yang digunakan yaitu sebanyak 300. Penjelasan nilai *loss*, *accuracy*, *val_loss* dan *val_accuracy* adalah sebagai berikut:

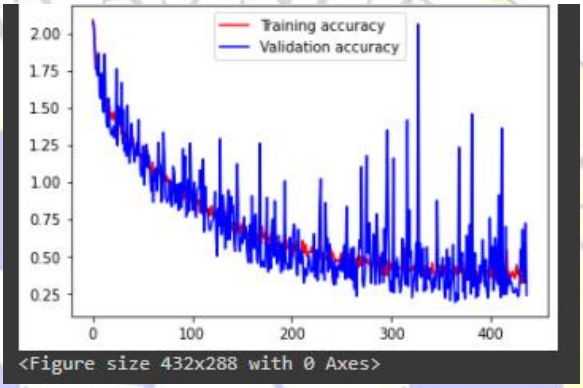
1. *Loss* menyatakan nilai dari loss function, terlihat bahwa pada **Gambar 4.6** *epoch* kesatu *loss* mendapatkan nilai yang cukup besar kemudian nilai itu menurun untuk setiap *epoch.*
2. *Accuracy* merupakan nilai akurasi dari data latih, terlihat pada **Gambar 4.6** akurasi yang didapat 0.1723 atau bisa dibilang 17%, hal ini diharapkan terus meningkat seiring bertambahnya iterasi.
3. *Val_loss,* menyatakan nilai dari loss function pada data validasi, terlihat bahwa pada **Gambar 4.6** *epoch* kesatu *loss* mendapatkan nilai yang cukup besar kemudian nilai itu fluktuatif untuk setiap *epoch.*
4. *Val_accuracy* sesuai namanya, *val* merupakan nilai akurasi dari data validasi, terlihat pada **Gambar 4.6** akurasi yang didapat 0.2688 atau bisa dibilang 26%, hal ini diharapkan terus meningkat seiring bertambahnya iterasi.

4.5. Hasil Pengujian Model

Pengujian model ini akan mengimplementasikan *callbacks* dan menggunakan jumlah *epoch* yang besar untuk mengetahui kinerja model. Pada implementasi *Callbacks* akan menargetkan nilai akurasi 90% dengan jumlah *epoch* 500.



Gambar 4.7 Grafik hasil *training accuracy* dan *validation accuracy*



Gambar 4.8 Grafik hasil *training loss* dan *validation loss*

```
24/24 - 65s - loss: 0.3663 - accuracy: 0.8932 - val_loss: 0.2516 - val_accuracy: 0.9414
Epoch 425/500
24/24 - 66s - loss: 0.3489 - accuracy: 0.8925 - val_loss: 0.2608 - val_accuracy: 0.9062
Epoch 426/500
24/24 - 66s - loss: 0.3977 - accuracy: 0.8885 - val_loss: 0.2831 - val_accuracy: 0.9180
Epoch 427/500
24/24 - 67s - loss: 0.4402 - accuracy: 0.8730 - val_loss: 0.2640 - val_accuracy: 0.9453
Epoch 428/500
24/24 - 66s - loss: 0.3795 - accuracy: 0.8785 - val_loss: 0.2269 - val_accuracy: 0.9453
Epoch 429/500
24/24 - 65s - loss: 0.3703 - accuracy: 0.8758 - val_loss: 0.2339 - val_accuracy: 0.9062
Epoch 430/500
24/24 - 66s - loss: 0.4047 - accuracy: 0.8698 - val_loss: 0.3488 - val_accuracy: 0.8867
Epoch 431/500
24/24 - 65s - loss: 0.3153 - accuracy: 0.8965 - val_loss: 0.5442 - val_accuracy: 0.8438
Epoch 432/500
24/24 - 66s - loss: 0.3747 - accuracy: 0.8818 - val_loss: 0.3941 - val_accuracy: 0.8828
Epoch 433/500
24/24 - 67s - loss: 0.3544 - accuracy: 0.8832 - val_loss: 0.6787 - val_accuracy: 0.8320
Epoch 434/500
24/24 - 66s - loss: 0.3687 - accuracy: 0.8825 - val_loss: 0.3188 - val_accuracy: 0.9062
Epoch 435/500
24/24 - 64s - loss: 0.5686 - accuracy: 0.8525 - val_loss: 0.3410 - val_accuracy: 0.8828
Epoch 436/500
24/24 - 65s - loss: 0.3322 - accuracy: 0.8919 - val_loss: 0.7190 - val_accuracy: 0.8242
Epoch 437/500
24/24 - 66s - loss: 0.3212 - accuracy: 0.9132 - val_loss: 0.2361 - val_accuracy: 0.9258

PELATIHAN BERHENTI, AKURASI MODEL SUDAH LEBIH DARI 90%!
```

Gambar 4.9 Proses pelatihan tiap *epoch*

Pada **Gambar 4.7**, **Gambar 4.8** grafik hasil pelatihan cukup baik nilai *accuracy* naik secara seimbang serta nilai *loss* turun dengan seimbang dan

Gambar 4.9 merupakan hasil proses pelatihan model terhadap jumlah *epoch*, dapat diketahui bahwa pada proses pengujian ini menargetkan akurasi menggunakan *callbacks* yakni 90%. Model berhasil mencapai nilai akurasi 91% pada *epoch* ke-437 dan proses pelatihan dihentikan secara otomatis.

Tabel 4.3 dibawah ini merupakan tabel hasil proses pelatihan menggunakan *epoch* 500 dilihat dari segi *training loss, training accuracy, validation loss, validation accuracy* dan *computation time* atau waktu komputasi per-*epoch.*

Tabel 4.3 Hasil proses *training* terhadap jumlah *epoch*

Epoch	***Training Loss***	***Training Accuracy***	***Validation Loss***	***Validation Accuracy***	**Waktu Komputasi (*second)***
437	0.3212	0.9132	0.2361	0.9258	64*s*

Tabel 4.3 dibawah ini merupakan hasil dari pengujian terhadap *training* dengan 500 *epoch*. Digunakan 20 gambar data uji dari data test pada masing-masing kelas untuk dilakukan pengujian.

Tabel 4.3 Hasil pengujian dengan data testing

Label	**Jumlah Data Uji**	**Prediksi Benar**	**Prediksi Salah**	**Akurasi**
Kue Cucur	20	13	7	65%
Kue Dadar Gulung	20	19	1	95%
Kue Klepon	20	19	1	95%
Kue Lapis	20	13	7	65%
Kue Lumpur	20	18	2	90%

Lanjutan **Tabel 4.3** Hasil pengujian dengan data testing

Label	Jumlah Data Uji	Prediksi Benar	Prediksi Salah	Akurasi
Kue Onde-onde	20	18	2	90%
Kue Risoles	20	16	4	80%
Kue Serabi	20	17	3	85%

4.6. Interpretasi Hasil

Pada **Bab 4.5 Hasil Pengujian** telah membahas pengujian model mengimplementasikan *epoch* yang besar dan menggunakan *callbacks*. Hal ini bertujuan untuk mengetahui kinerja model dan mencari model yang ideal. Hasil terbaik pada pengujian menggunakan data testing mendapat nilai akurasi sebesar 95% yaitu kue dadar gulung dan klepon. Sedangkan hasil terkecil mendapat nilai akurasi sebesar 65% yaitu kue lapis dan kue cucur.