



PICTURES CLASSIFICATION CAT VS DOG

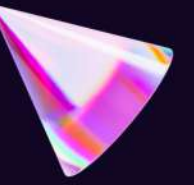
FARHANUL KHAIR
2208107010076



DATASET YANG DIGUNAKAN



Pada klasifikasi kali ini saya menggunakan dataset **CIFAR-10** adalah dataset gambar yang sering digunakan dalam penelitian visi komputer dan pembelajaran mesin. Dataset ini terdiri dari 60.000 gambar berwarna berukuran 32x32 piksel, yang terbagi menjadi 10 kelas berbeda seperti pesawat, mobil, burung, kucing, rusa, anjing, katak, kuda, kapal, dan truk. Dataset ini memiliki 50.000 gambar untuk pelatihan dan 10.000 gambar untuk pengujian, menjadikannya ideal untuk mengembangkan dan menguji algoritma klasifikasi gambar.



Dataset CIFAR-10 ini disediakan oleh framework Tensorflow, jadi jika ingin menggunakan dataset ini tidak perlu untuk mengunduh manual. Cara aksesnya juga mudah hanya dengan melalui modul `tensorflow.keras.datasets`.

Lalu dari dataset ini saya hanya memilih data dengan label kelas Kucing dan Anjing saja.





JUMLAH FITUR DAN LABEL

Informasi total data yaitu fitur dan label

- Fitur: Ada 60.000 gambar secara total, yang dibagi menjadi 40.000 untuk pelatihan, 10.000 untuk validasi, dan 10.000 untuk pengujian.
- Label: Ada 60.000 label yang sesuai dengan gambar-gambar tersebut.

Namun karena yang saya gunakan adalah hanya data Kucing dan Anjing jadi total data fitur dan label nya adalah :

- Kucing 3977
- Anjing 3985



JENIS JARINGAN SARAF YANG DIGUNAKAN

Convolutional Neural Network (CNN) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk memproses data seperti gambar. CNN bekerja dengan menggunakan lapisan konvolusi untuk mengekstrak fitur penting dari gambar, seperti tepi atau tekstur, diikuti oleh lapisan pooling untuk mengurangi dimensi data sambil mempertahankan informasi utama. Setelah itu, data diratakan melalui lapisan fully connected untuk membuat prediksi, misalnya klasifikasi gambar. CNN efektif karena dapat mengenali pola lokal dan global dengan parameter yang lebih efisien, menjadikannya sangat cocok untuk tugas seperti klasifikasi, deteksi objek, dan segmentasi gambar.

JENIS OPTIMASI YANG DIGUNAKAN

ADAM OPTIMIZER

Adam adalah metode pengoptimalan berbasis gradien yang menggunakan estimasi momen pertama (mean) dan kedua (variance) untuk mempercepat konvergensi.

JENIS FUNGSI AKTIVASI YANG DIGUNAKAN

01

CONV2D LAYERS:
RELU (RECTIFIED
LINEAR UNIT)

Fungsi aktivasi ini umum digunakan karena memperkenalkan non-linearitas tanpa masalah vanishing gradient.

02

OUTPUT LAYER:
SIGMOID

Fungsi sigmoid digunakan karena output adalah klasifikasi biner (anjing vs. kucing), yang memberikan nilai probabilitas antara 0 dan 1.

JUMLAH HIDDEN LAYER

HIDDEN LAYERS = 4

2 lapisan konvolusi: Conv2D (32 filter, 3x3 kernel), Conv2D (64 filter, 3x3 kernel).

2 lapisan pooling: MaxPooling2D (2x2 pooling).



JUMLAH TOTAL HIDDEN NODE PER LAYER

HIDDEN NODES :

CONV2D (32 FILTERS)

Output dimensi adalah
(32x32x3) → (30x30x32)
setelah operasi
konvolusi.

MAXPOOLING2D

Setelah pooling,
dimensi menjadi
(15x15x32).

CONV2D (64 FILTERS)

Output dimensi setelah
konvolusi menjadi
(13x13x64).

MAXPOOLING2D

Setelah pooling,
dimensi menjadi
(6x6x64).

Total nodes adalah hasil pengalihan dimensi akhir lapisan sebelum flattening.



JUMLAH TOTAL BOBOT (WEIGHT)

Jumlah bobot (parameter) dapat
dihitung berdasarkan kernel, filter, dan
bias di setiap lapisan.

Model: "sequential"

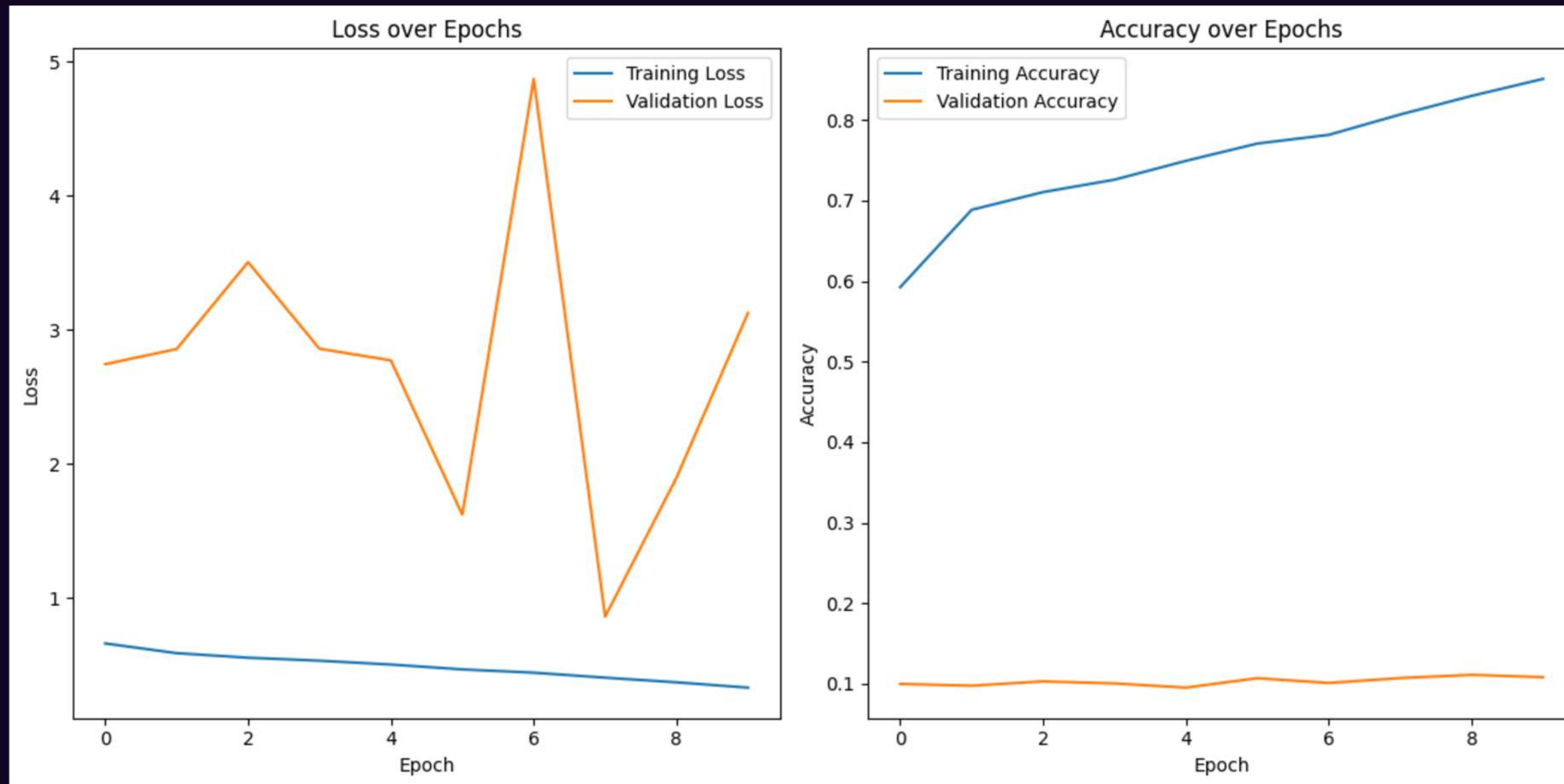
| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d (Conv2D) | (None, 30, 30, 32) | 896 |
| max_pooling2d (MaxPooling2D) | (None, 15, 15, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 13, 13, 64) | 18,496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 6, 6, 64) | 0 |
| flatten (Flatten) | (None, 2304) | 0 |
| dense (Dense) | (None, 64) | 147,520 |
| dense_1 (Dense) | (None, 1) | 65 |

Total params: 166,977 (652.25 KB)

Trainable params: 166,977 (652.25 KB)

Non-trainable params: 0 (0.00 B)

PLOT PROSES TRAINING



AKURASI YANG DIHASILKAN

```
63/63 - 0s - 6ms/step - accuracy: 0.7500 - loss: 0.5526  
Test Loss: 0.5526  
Test Accuracy: 0.7500
```


EVENT TENSORBOARD

```
Epoch 1/10
249/249 ————— 6s 18ms/step - accuracy: 0.5911 - loss: 0.6670 - val_accuracy: 0.0963 - val_loss: 2.6597
Epoch 2/10
249/249 ————— 4s 16ms/step - accuracy: 0.6663 - loss: 0.6024 - val_accuracy: 0.1016 - val_loss: 4.2341
Epoch 3/10
249/249 ————— 4s 14ms/step - accuracy: 0.6986 - loss: 0.5651 - val_accuracy: 0.1054 - val_loss: 0.7891
Epoch 4/10
249/249 ————— 4s 14ms/step - accuracy: 0.7397 - loss: 0.5166 - val_accuracy: 0.0988 - val_loss: 6.1754
Epoch 5/10
249/249 ————— 4s 15ms/step - accuracy: 0.7629 - loss: 0.4878 - val_accuracy: 0.1045 - val_loss: 1.7964
Epoch 6/10
249/249 ————— 4s 14ms/step - accuracy: 0.7784 - loss: 0.4594 - val_accuracy: 0.1095 - val_loss: 2.7325
Epoch 7/10
249/249 ————— 4s 15ms/step - accuracy: 0.7998 - loss: 0.4281 - val_accuracy: 0.1040 - val_loss: 5.0500
Epoch 8/10
249/249 ————— 4s 15ms/step - accuracy: 0.8129 - loss: 0.3975 - val_accuracy: 0.1033 - val_loss: 5.6853
Epoch 9/10
249/249 ————— 4s 15ms/step - accuracy: 0.8331 - loss: 0.3662 - val_accuracy: 0.1052 - val_loss: 2.5303
Epoch 10/10
249/249 ————— 4s 15ms/step - accuracy: 0.8523 - loss: 0.3328 - val_accuracy: 0.1106 - val_loss: 2.3220
```


THANK YOU <3