

# APLIKASI KLASIFIKASI GAMBAR DENGAN MODEL TRANSFER LEARNING

---

Presented by Kelompok 7

---

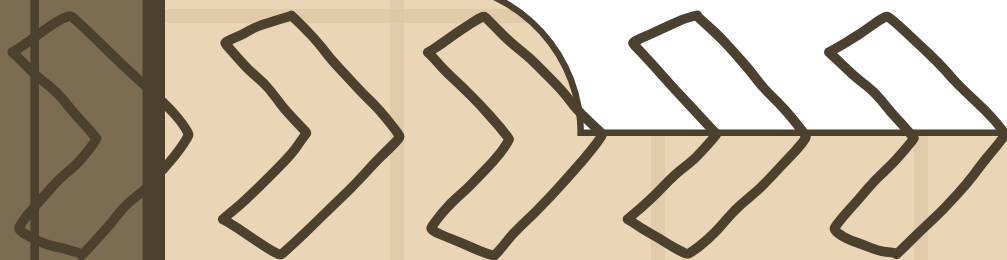
Akhiril Anwar Harahap

Bayu Delvika

Kurnyadi Dwi Putra

Putri Ardina

Tita Alisya



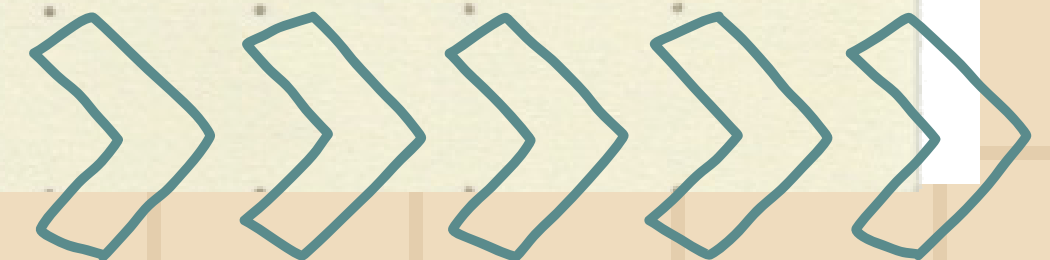


# Latar Belakang Masalah

Salah satu domain dalam Artificial Intelligence adalah **Computer vision**. Computer vision merupakan salah satu domain yang cukup menarik sekaligus menantang untuk dipelajari. Salah satu contoh dari penerapan computer vision dalam kehidupan sehari-hari adalah **klasifikasi gambar**

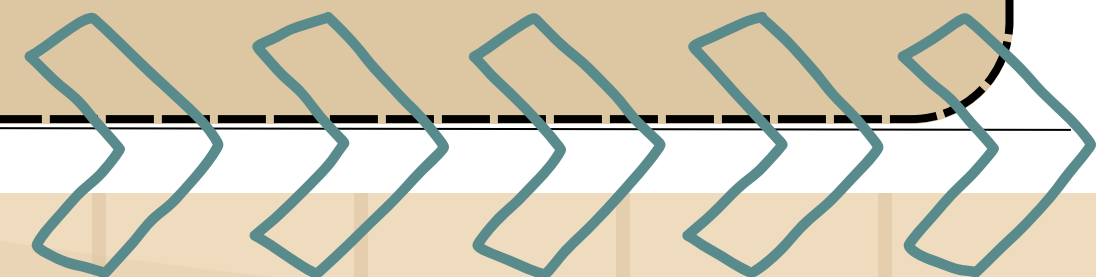
Klasifikasi gambar adalah suatu jenis tugas dalam bidang computer vision yang bertujuan untuk mengelompokkan gambar ke dalam kategori atau label tertentu berdasarkan karakteristik atau fitur yang terdapat pada gambar tersebut.

Pada kesempatan kali ini, kelompok 7 akan membahas tentang penerapan dari computer vision, yakni **aplikasi klasifikasi gambar pada dataset cifar10**.



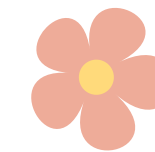
# Rumusan Masalah

1. Apa itu cifar10
2. Model apa saja yang digunakan untuk melakukan klasifikasi gambar?
3. Bagaimana perbandingan dari hasil evaluasi tiap-tiap model?
4. Bagaimana Flowchart sistem yang dibangun?
5. Bagaimana arsitektur sistem yang dibangun?
6. Bagaimana user interface dari aplikasi yang dibangun?
7. Bagaimana analisa dan kesimpulan yang bisa diperoleh?





# Teori Cifar 10



Cifar10 adalah dataset yang sering digunakan untuk melatih model dalam pengenalan suatu gambar. Cifar10 ini memiliki 60.000 gambar berwarna dengan resolusi 32x32 pixel dengan di bagi menjadi 10 kelas. Dengan menggunakan cifar ini kita dapat memanfaatkan pengetahuan yang telah ada di dalam suatu model yang telah di latih sebelumnya dan dapat menggunakan nya sebagai titik awal yang baik untuk meningkatkan kinerja model pada tugas pengenalan gambar yang baru



# Model yang digunakan

## Model Convolutional Neural Network (CNN)

Model CNN (Convolutional Neural Network) adalah tipe arsitektur jaringan saraf tiruan yang biasanya digunakan dalam tugas-tugas pengolahan citra, seperti klasifikasi gambar, deteksi objek, dan segmentasi gambar



# Model yang digunakan

## ResNet (Residual Network)

ResNet adalah arsitektur ini diperkenalkan pada tahun 2015 oleh Kaiming He dan rekan-rekannya.

ResNet mengatasi masalah ini dengan menggunakan modul "residual" yang memungkinkan informasi asli pada lapisan sebelumnya dapat dijaga dan digunakan dalam lapisan selanjutnya. Dalam hal ini, setiap lapisan mempelajari perubahan residual terhadap lapisan sebelumnya, sehingga memudahkan jaringan untuk belajar representasi yang lebih baik.

## VGG (Visual Geometry Group)

VGG adalah arsitektur jaringan saraf konvolusi yang terdiri dari beberapa lapisan konvolusi berukuran kecil ( $3 \times 3$ ) yang diikuti oleh lapisan penggabungan (pooling) dan lapisan fully connected (FC).

VGG memiliki kelebihan dalam mempelajari fitur lokal dalam citra dengan baik dan memberikan representasi yang kuat.





# Arsitektur Model

CNN tanpa pre-trained model

```
def make_model():  
    model = Sequential()  
    model.add(Conv2D(16, (3, 3), input_shape=(32, 32, 3), padding='same'))  
    model.add(LeakyReLU(0.1))  
    model.add(Conv2D(32, (3, 3), padding='same'))  
    model.add(LeakyReLU(0.1))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(Dropout(0.25))  
    model.add(Conv2D(32, (3, 3), padding='same'))  
    model.add(LeakyReLU(0.1))  
    model.add(Conv2D(64, (3, 3), padding='same'))  
    model.add(LeakyReLU(0.1))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(Dropout(0.25))  
    model.add(Flatten())  
    model.add(Dense(256))  
    model.add(LeakyReLU(0.1))  
    model.add(Dropout(0.5))  
    model.add(Dense(10, activation='softmax'))  
  
    return model
```



# Arsitektur Model

## Resnet50

```
def make_model():  
    base_model_1 = tf.keras.applications.resnet50.ResNet50(include_top=False, weights='imagenet',  
                                                            input_shape=(32, 32, 3), classes=10)  
  
    model = Sequential()  
    model.add(base_model_1)  
    model.add(Flatten())  
    model.add(Dense(1024, activation='relu', input_dim=512))  
    model.add(Dense(512, activation='relu'))  
    model.add(Dense(256, activation='relu'))  
    model.add(Dense(128, activation='relu'))  
    model.add(Dense(10, activation='softmax'))  
    return model
```



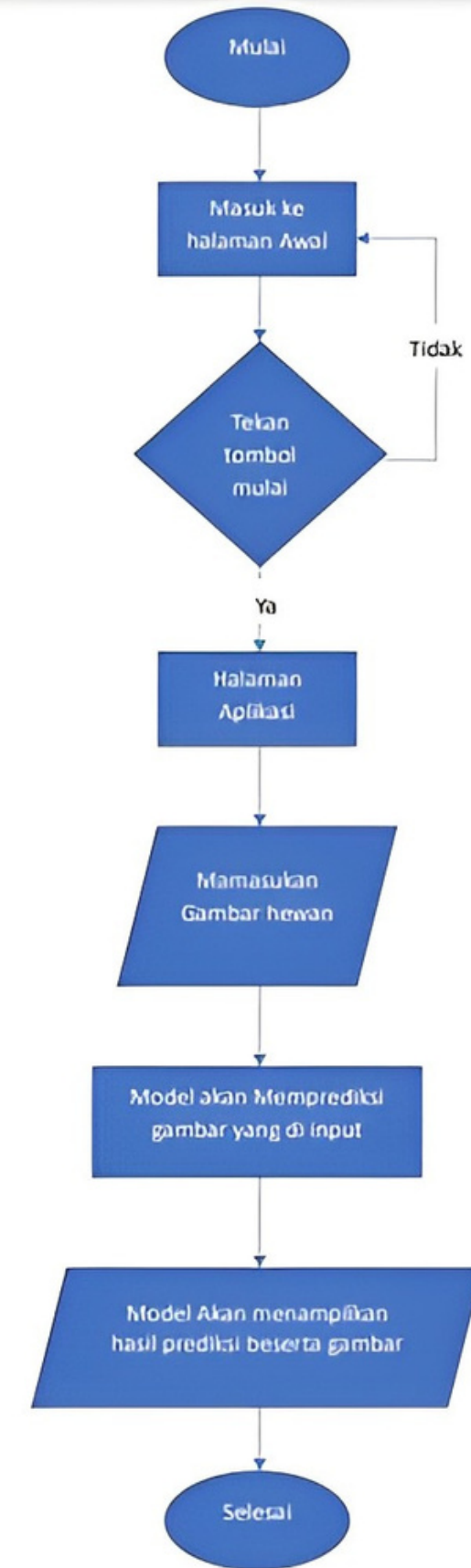
# Arsitektur Model

## VGG 19

```
[ ] def make_model():
    base_model_1 = tf.keras.applications.vgg19.VGG19(include_top=False, weights='imagenet',
                                                    input_shape=(32, 32, 3), classes=10)

    model = Sequential()
    model.add(base_model_1)
    model.add(Flatten())
    model.add(Dense(1024, activation='relu', input_dim=512))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(256, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    return model
```

# Flowchart



# Perbandingan Akurasi Model

CNN tanpa pre-trained model

```
Epoch 1/10
1563/1563 [=====] - 165s 105ms/step - loss: 1.6475 - accuracy: 0.3934 - val_loss: 1.0107 - val_accuracy: 0.6429
Epoch 2/10
1563/1563 [=====] - 150s 96ms/step - loss: 1.0333 - accuracy: 0.6360 - val_loss: 0.9045 - val_accuracy: 0.6887
Epoch 3/10
1563/1563 [=====] - 150s 96ms/step - loss: 0.8719 - accuracy: 0.6940 - val_loss: 0.7842 - val_accuracy: 0.7258
Epoch 4/10
1563/1563 [=====] - 150s 96ms/step - loss: 0.7732 - accuracy: 0.7304 - val_loss: 0.7281 - val_accuracy: 0.7455
Epoch 5/10
1563/1563 [=====] - 147s 94ms/step - loss: 0.7278 - accuracy: 0.7470 - val_loss: 0.6896 - val_accuracy: 0.7607
Epoch 6/10
1563/1563 [=====] - 162s 103ms/step - loss: 0.6683 - accuracy: 0.7674 - val_loss: 0.6940 - val_accuracy: 0.7603
Epoch 7/10
1563/1563 [=====] - 142s 91ms/step - loss: 0.6372 - accuracy: 0.7756 - val_loss: 0.6595 - val_accuracy: 0.7676
Epoch 8/10
1563/1563 [=====] - 135s 86ms/step - loss: 0.6085 - accuracy: 0.7876 - val_loss: 0.6494 - val_accuracy: 0.7765
Epoch 9/10
1563/1563 [=====] - 140s 90ms/step - loss: 0.5842 - accuracy: 0.7941 - val_loss: 0.6217 - val_accuracy: 0.7832
Epoch 10/10
1563/1563 [=====] - 142s 91ms/step - loss: 0.5646 - accuracy: 0.8031 - val_loss: 0.6579 - val_accuracy: 0.7859
```

# Perbandingan Akurasi Model

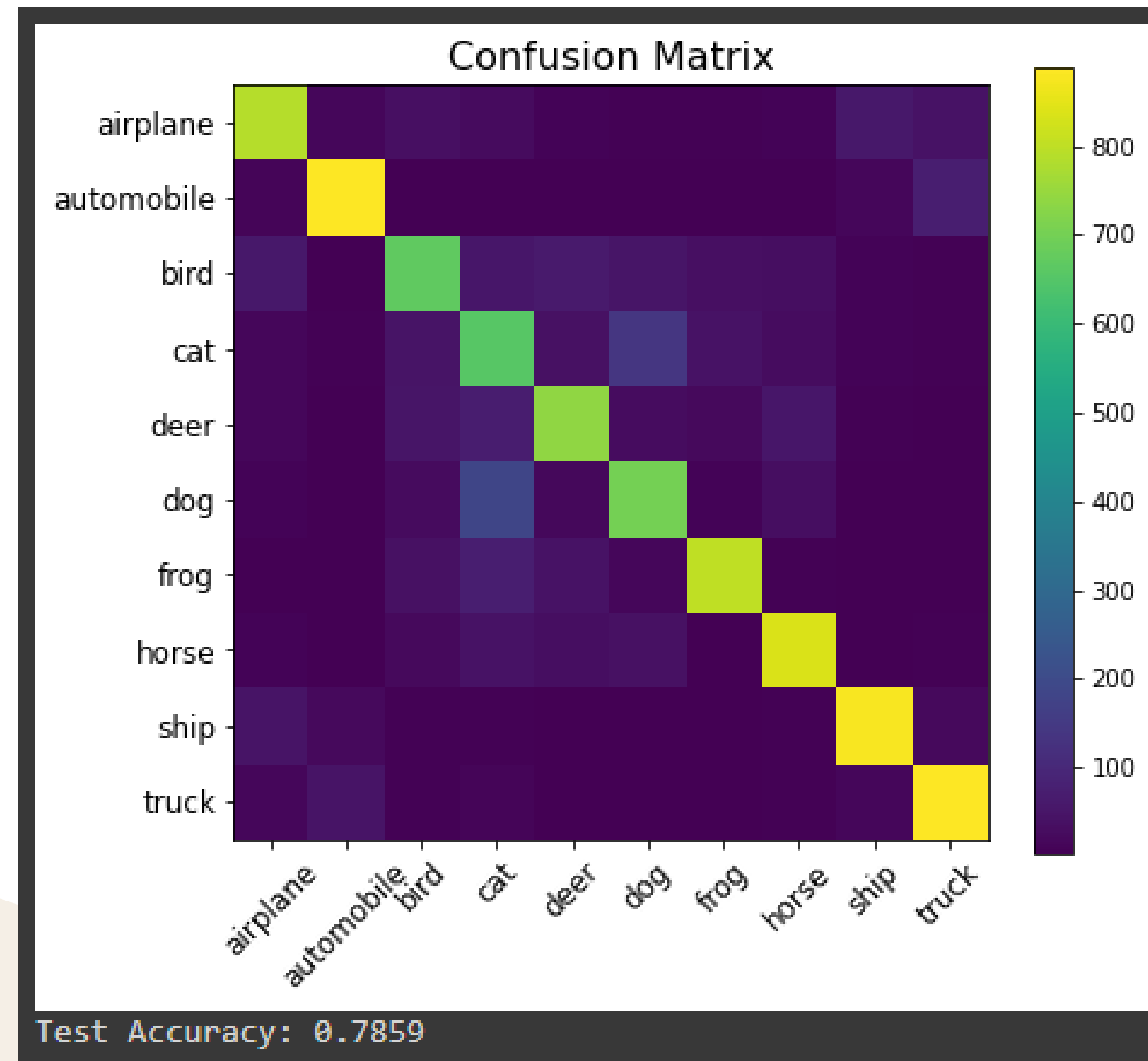
## Resnet

```
Epoch 35/50
350/350 [=====] - 44s 125ms/step - loss: 1.0458 - acc: 0.6851 - val_loss: 1.0308 - val_acc: 0.6692
Epoch 36/50
350/350 [=====] - 43s 124ms/step - loss: 1.0342 - acc: 0.6879 - val_loss: 1.0512 - val_acc: 0.6679
Epoch 37/50
350/350 [=====] - 44s 125ms/step - loss: 1.0184 - acc: 0.6948 - val_loss: 1.0309 - val_acc: 0.6695
Epoch 38/50
350/350 [=====] - 44s 125ms/step - loss: 1.0202 - acc: 0.6911 - val_loss: 1.0009 - val_acc: 0.6791
Epoch 39/50
350/350 [=====] - 43s 124ms/step - loss: 1.0055 - acc: 0.6967 - val_loss: 1.0076 - val_acc: 0.6788
Epoch 40/50
350/350 [=====] - 44s 126ms/step - loss: 0.9789 - acc: 0.6966 - val_loss: 1.0004 - val_acc: 0.6768
Epoch 41/50
350/350 [=====] - 43s 123ms/step - loss: 0.9338 - acc: 0.7032 - val_loss: 0.9834 - val_acc: 0.6856
Epoch 42/50
350/350 [=====] - 43s 123ms/step - loss: 0.9160 - acc: 0.7088 - val_loss: 0.9575 - val_acc: 0.6866
Epoch 43/50
350/350 [=====] - 43s 122ms/step - loss: 0.8873 - acc: 0.7092 - val_loss: 0.9499 - val_acc: 0.6927
Epoch 44/50
350/350 [=====] - 42s 121ms/step - loss: 0.8454 - acc: 0.7172 - val_loss: 0.9367 - val_acc: 0.6968
Epoch 45/50
350/350 [=====] - 42s 120ms/step - loss: 0.8377 - acc: 0.7167 - val_loss: 0.9288 - val_acc: 0.6970
Epoch 46/50
350/350 [=====] - 42s 120ms/step - loss: 0.8190 - acc: 0.7245 - val_loss: 0.9230 - val_acc: 0.6999
Epoch 47/50
350/350 [=====] - 42s 119ms/step - loss: 0.8022 - acc: 0.7294 - val_loss: 0.9097 - val_acc: 0.7046
Epoch 48/50
350/350 [=====] - 42s 119ms/step - loss: 0.7944 - acc: 0.7306 - val_loss: 0.8917 - val_acc: 0.7130
Epoch 49/50
350/350 [=====] - 42s 121ms/step - loss: 0.7720 - acc: 0.7401 - val_loss: 0.8725 - val_acc: 0.7131
Epoch 50/50
350/350 [=====] - 42s 120ms/step - loss: 0.7638 - acc: 0.7432 - val_loss: 0.8722 - val_acc: 0.7185
<keras.callbacks.History at 0x79e03ee0cb70>
```

## VGG

```
Epoch 35/50
350/350 [=====] - 28s 81ms/step - loss: 0.0950 - acc: 0.9698 - val_loss: 0.5279 - val_acc: 0.8564
Epoch 36/50
350/350 [=====] - 28s 80ms/step - loss: 0.0935 - acc: 0.9698 - val_loss: 0.5408 - val_acc: 0.8555
Epoch 37/50
350/350 [=====] - 28s 80ms/step - loss: 0.0966 - acc: 0.9688 - val_loss: 0.5286 - val_acc: 0.8576
Epoch 38/50
350/350 [=====] - 28s 80ms/step - loss: 0.0930 - acc: 0.9696 - val_loss: 0.5500 - val_acc: 0.8539
Epoch 39/50
350/350 [=====] - 28s 81ms/step - loss: 0.0921 - acc: 0.9708 - val_loss: 0.5499 - val_acc: 0.8538
Epoch 40/50
350/350 [=====] - 28s 80ms/step - loss: 0.0910 - acc: 0.9714 - val_loss: 0.5483 - val_acc: 0.8549
Epoch 41/50
350/350 [=====] - 28s 81ms/step - loss: 0.0899 - acc: 0.9719 - val_loss: 0.5348 - val_acc: 0.8581
Epoch 42/50
350/350 [=====] - 28s 81ms/step - loss: 0.0887 - acc: 0.9719 - val_loss: 0.5483 - val_acc: 0.8559
Epoch 43/50
350/350 [=====] - 27s 78ms/step - loss: 0.0874 - acc: 0.9726 - val_loss: 0.5548 - val_acc: 0.8558
Epoch 44/50
350/350 [=====] - 28s 81ms/step - loss: 0.0880 - acc: 0.9725 - val_loss: 0.5472 - val_acc: 0.8559
Epoch 45/50
350/350 [=====] - 28s 80ms/step - loss: 0.0870 - acc: 0.9719 - val_loss: 0.5544 - val_acc: 0.8556
Epoch 46/50
350/350 [=====] - 28s 81ms/step - loss: 0.0875 - acc: 0.9726 - val_loss: 0.5544 - val_acc: 0.8542
Epoch 47/50
350/350 [=====] - 28s 81ms/step - loss: 0.0845 - acc: 0.9732 - val_loss: 0.5498 - val_acc: 0.8574
Epoch 48/50
350/350 [=====] - 28s 81ms/step - loss: 0.0854 - acc: 0.9730 - val_loss: 0.5602 - val_acc: 0.8549
Epoch 49/50
350/350 [=====] - 28s 81ms/step - loss: 0.0825 - acc: 0.9743 - val_loss: 0.5515 - val_acc: 0.8553
Epoch 50/50
350/350 [=====] - 28s 81ms/step - loss: 0.0841 - acc: 0.9734 - val_loss: 0.5629 - val_acc: 0.8573
<keras.callbacks.History at 0x79e450c8aeb8>
```

# Perbandingan Akurasi Model



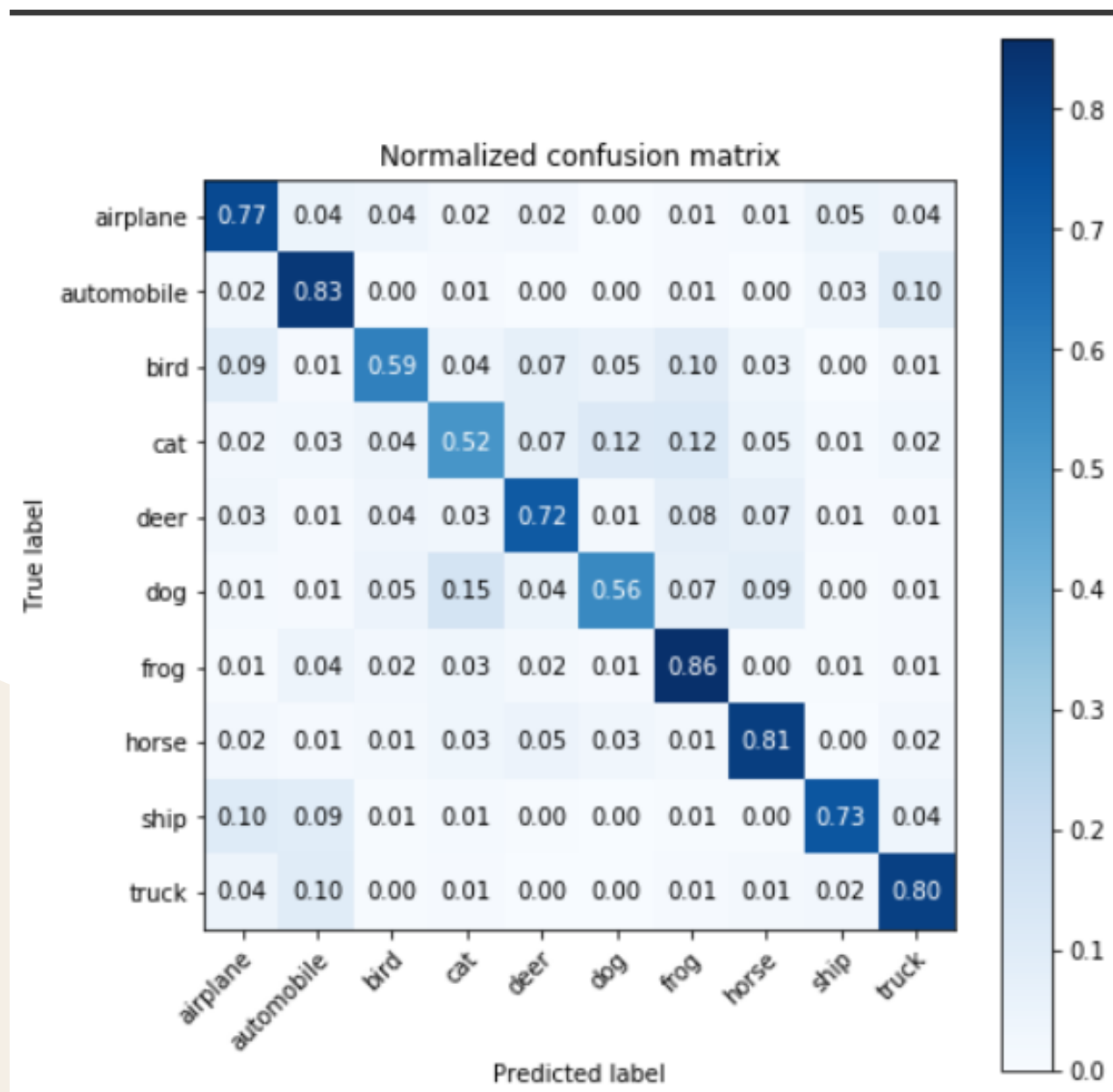
CNN tanpa pre-trained model

Akurasi 79%

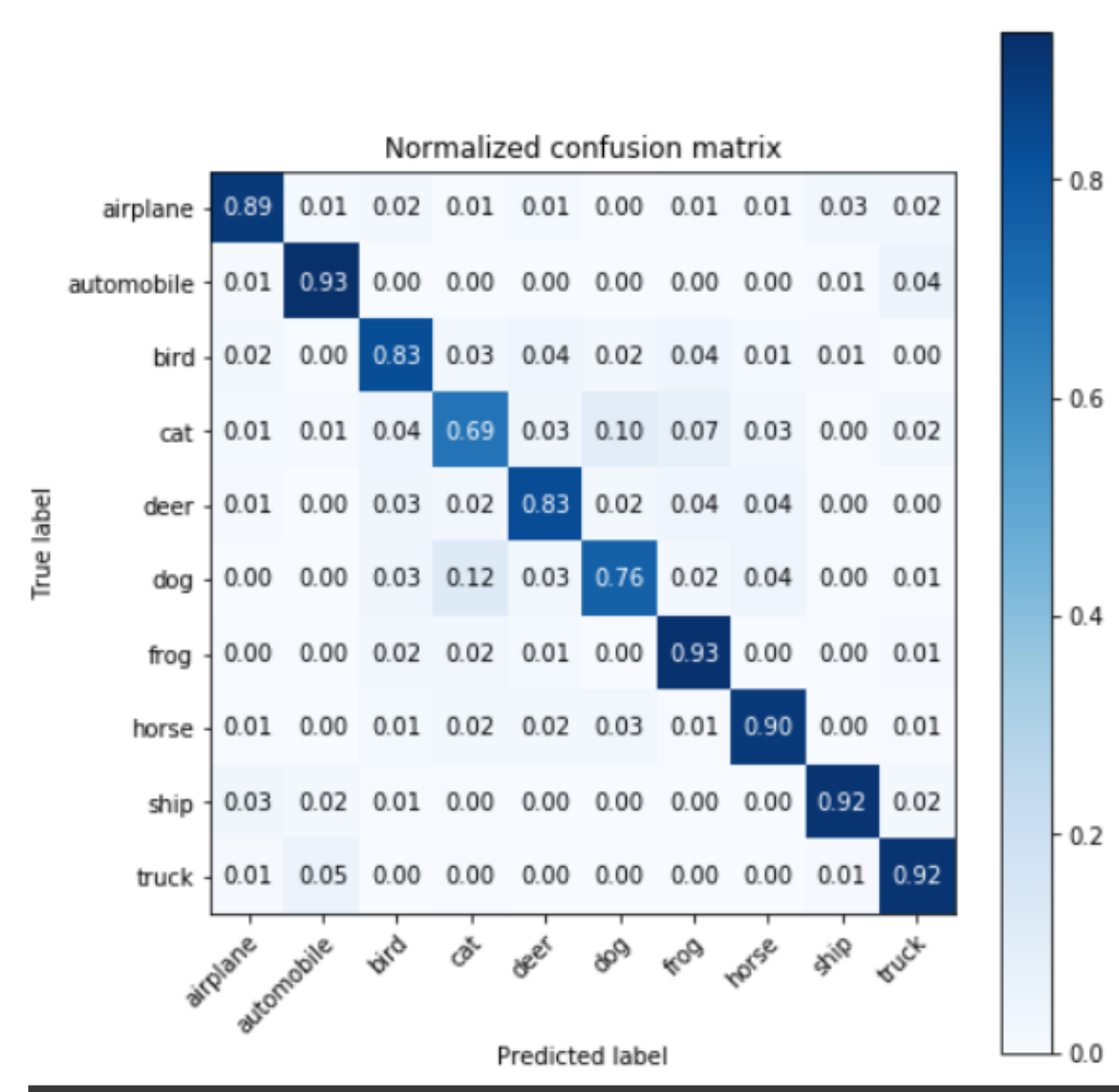
# Perbandingan Akurasi Model

## Resnet

## VGG



Akurasi 71%



Akurasi 86%





# Arsitektur Sistem

Sistem dibangun menggunakan framework flask. Flask adalah sebuah framework atau kerangka kerja web yang dibuat dengan bahasa pemrograman Python. Flask digunakan untuk membangun aplikasi web dengan cara yang sederhana dan fleksibel.

- static
- templates
- app
- fungsi
- model\_cifar10\_cnn\_tf

# User Interface (Halaman Informasi)



Orbit Future Academy

INFORMASI

APLIKASI

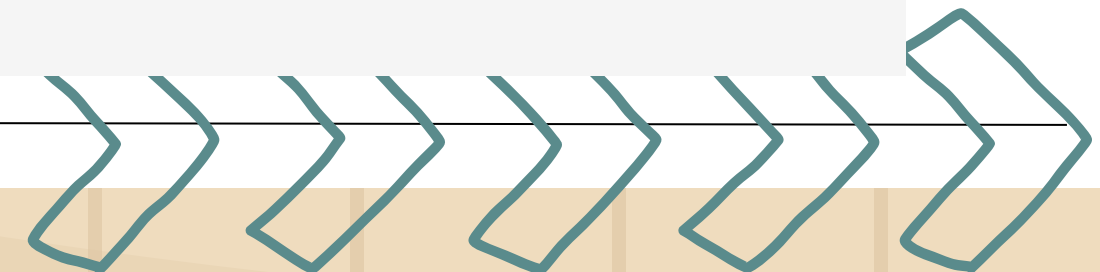
ANGGOTA KELOMPOK

## KELOMPOK

### Animal Image Classification


Aplikasi ini merupakan aplikasi yang dapat mengklasifikasikan gambar. Adapun dataset yang digunakan untuk model klasifikasi gambar ini adalah CIFAR10. Pada dataset ini, terdapat 10 objek yang dapat diklasifikasi yaitu pesawat, mobil, burung, kucing, rusa, anjing, katak, kuda, kapal, dan truk. Untuk dapat menggunakan aplikasi ini, pengguna dapat mengunggah sebuah gambar dengan format 'JPG', 'JPEG', dan 'PNG'. Hasil deteksi objek akan ditampilkan setelah pengguna mengunggah gambar tersebut.

Mulai



# User Interface (Halaman Informasi)





**Orbit Future Academy**

INFORMASI

APLIKASI

ANGGOTA KELOMPOK

Upload Gambar

**Kucing**

silahkan upload sebuah gambar

Drop files here


or

Browse...

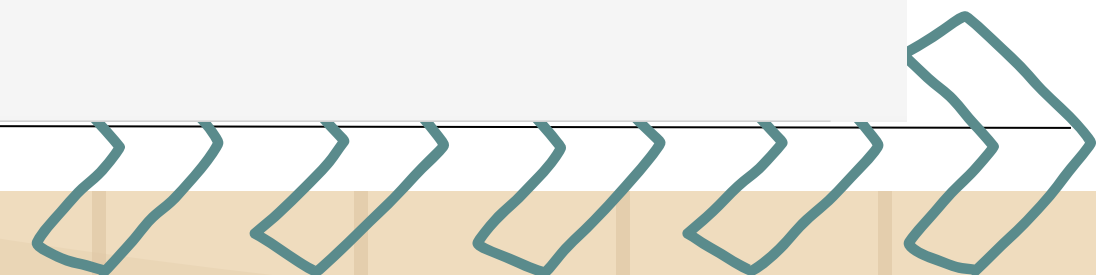
katak.jpg

Prediksi

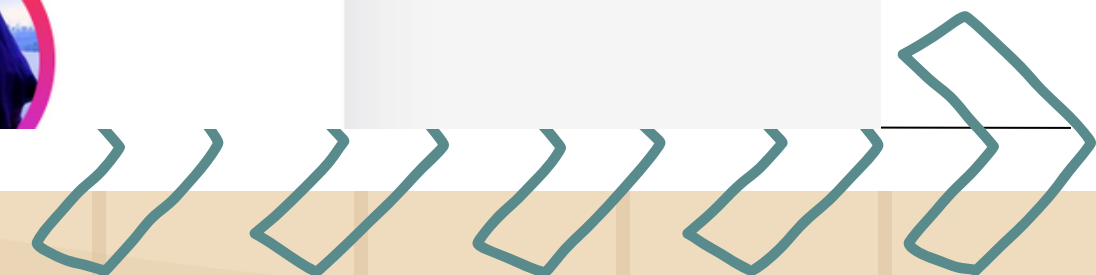
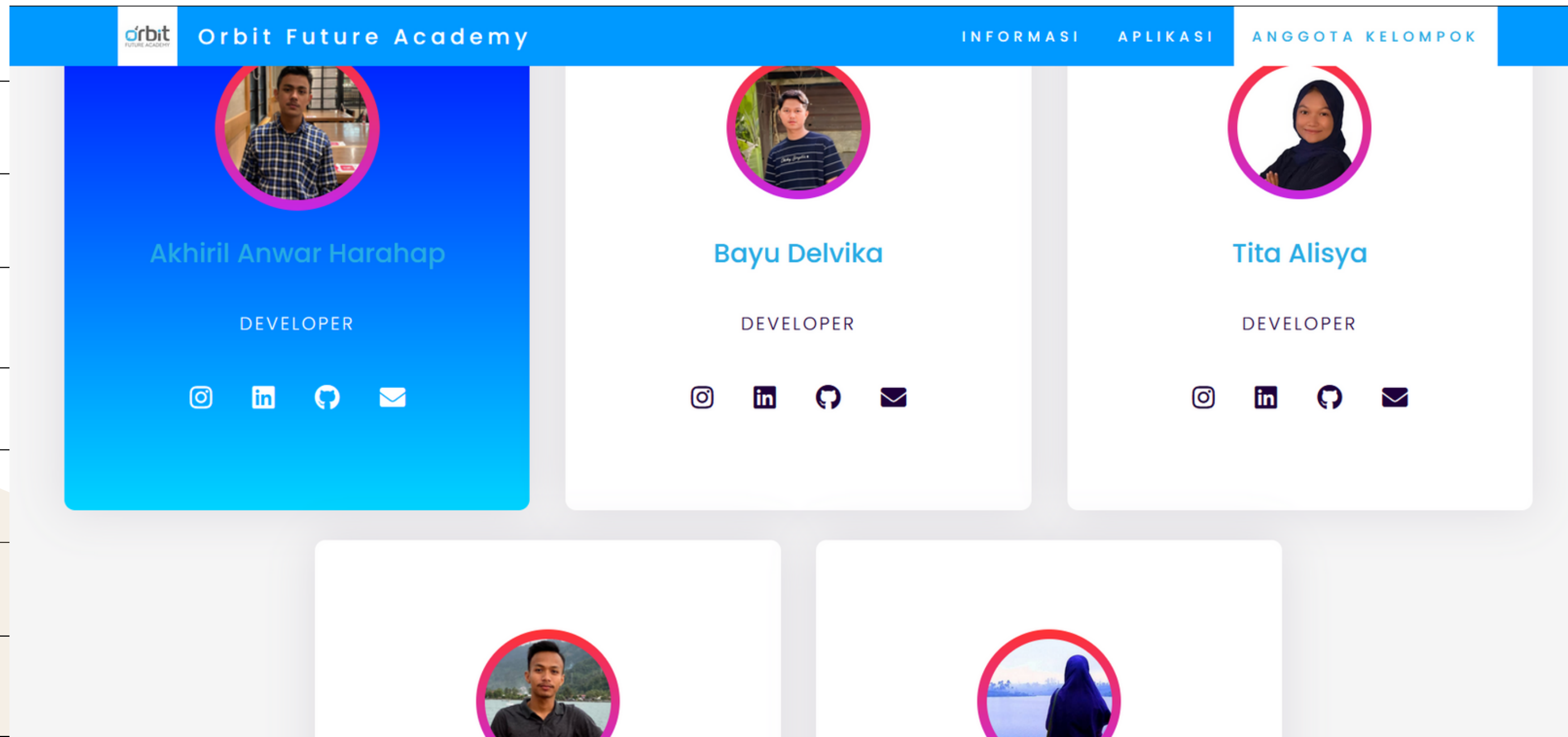
Hasil Prediksi



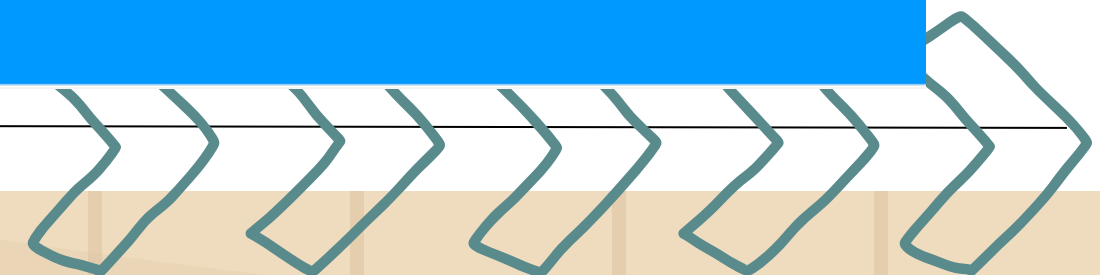
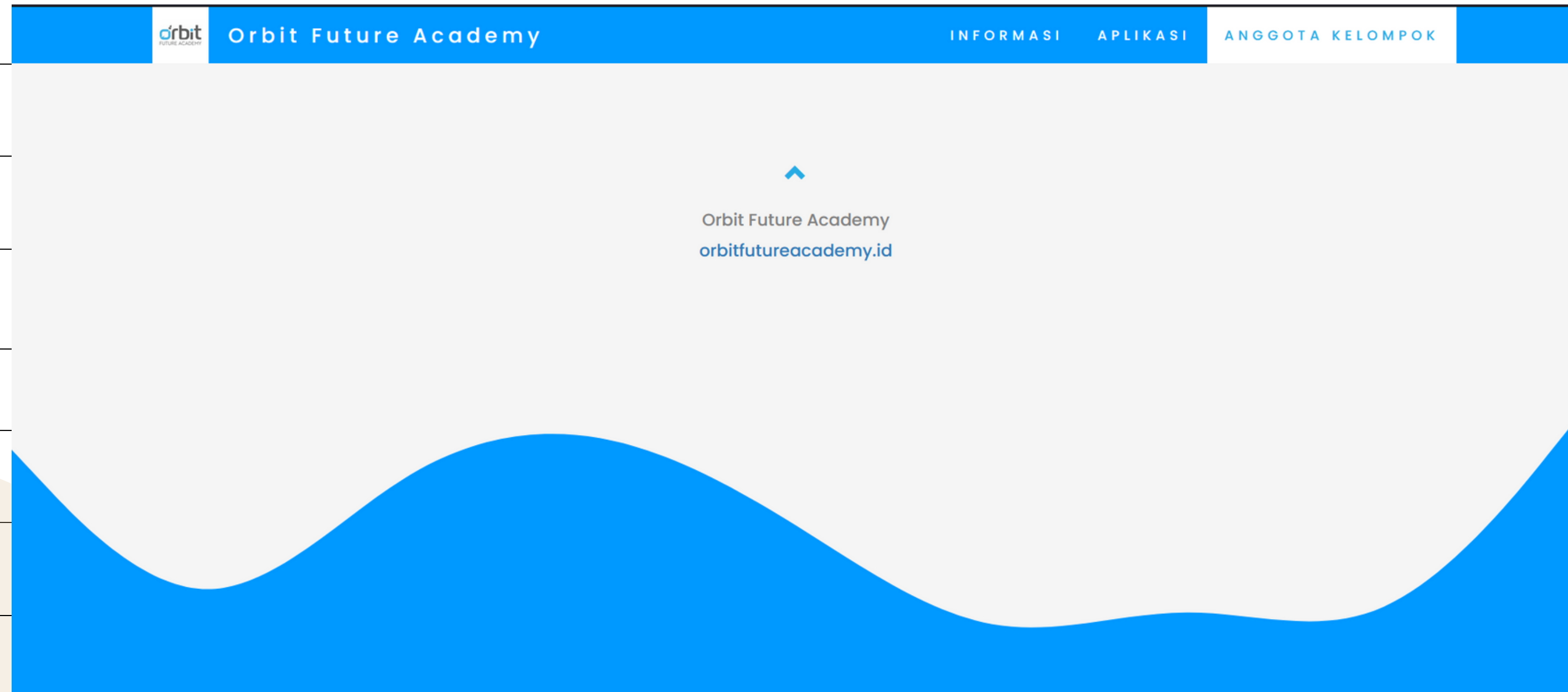
frog



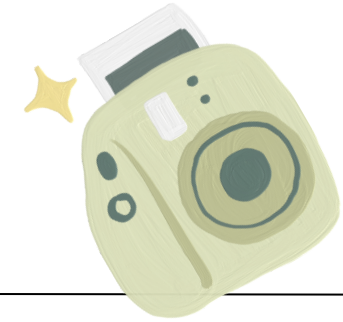
# User Interface (Halaman Informasi)



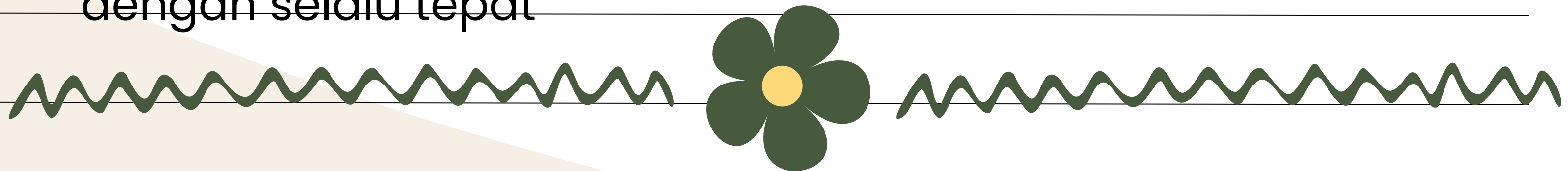
# User Interface (Halaman Informasi)



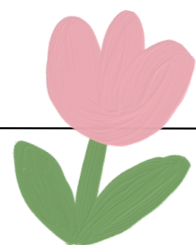
# Kesimpulan & Analisa



1. Dari 3 model yang dibangun, model dengan transfer learning **VGG19** merupakan model terbaik dengan **akurasi 86%**
2. VGG19 lebih baik dari Resnet50 sebab dataset cifar10 hanya berukuran 32x32px. Dimana Resnet50 terlalu overkill untuk dataset ini
3. Dari banyak percobaan, Aplikasi mampu melakukan klasifikasi dengan selalu tepat







THANK YOU

