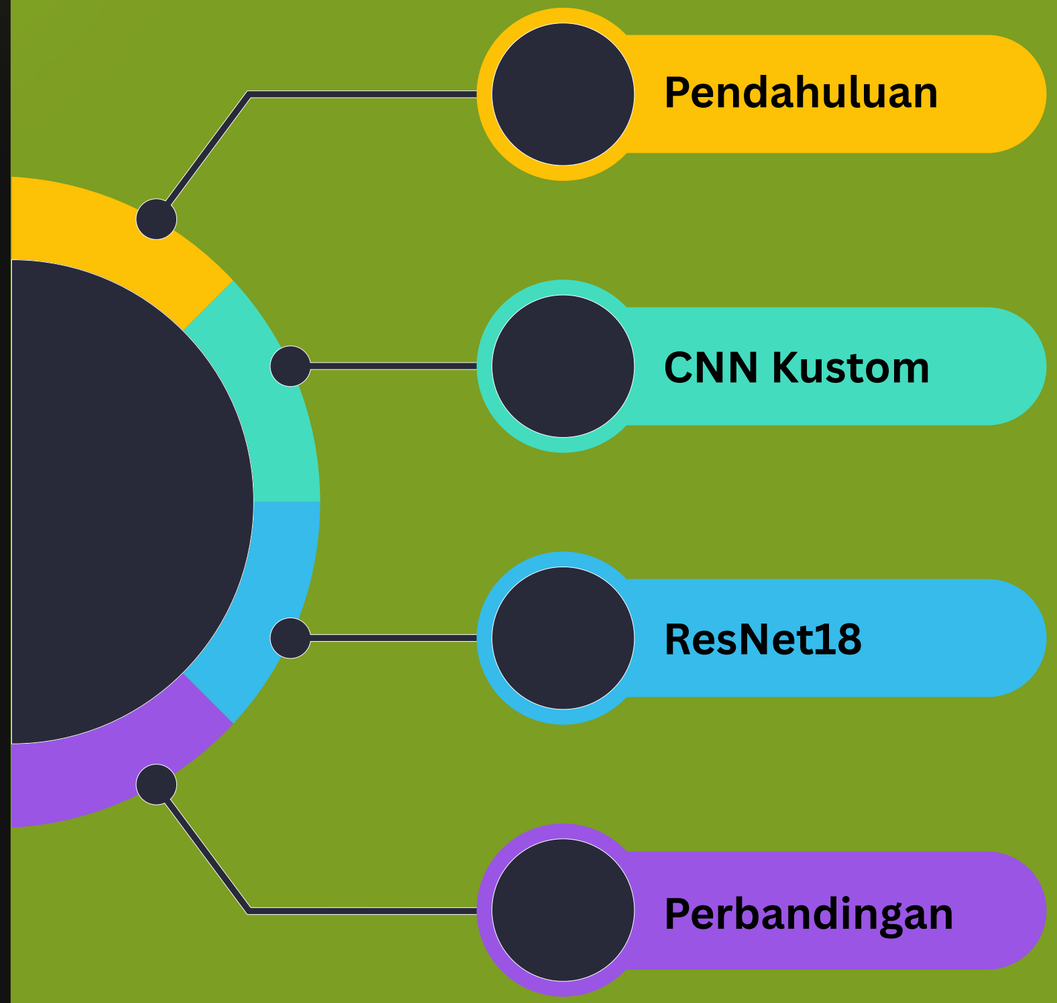
# CNN VS RESNET18

FOR CIFAR-10 CLASSIFICATION

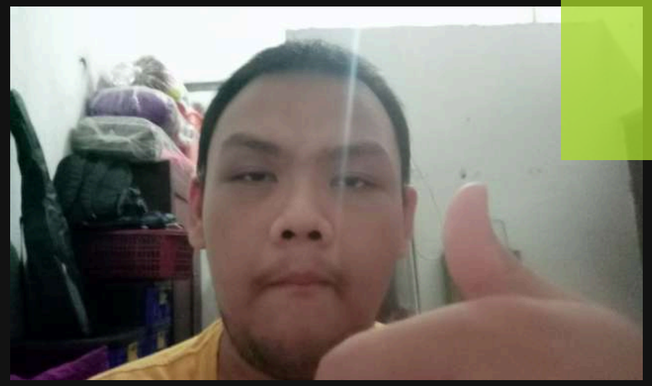
START SLIDE

# TABLE OF CONTENT



# MEMBER OF GROUP

**Alif Alamsyah**
**11220940000028**



**Ibnullabib**
**11220940000037**

# TUJUAN PROJEK

## 01.

### Klasifikasi CIFAR-10

Mengklasifikasi dataset CIFAR-10 dengan menggunakan model CNN kustom dan model ResNet18 pretrained yang sudah disesuaikan untuk dataset CIFAR-10 dari modul PyTorch

## 02.

### Hyperparameter Tuning

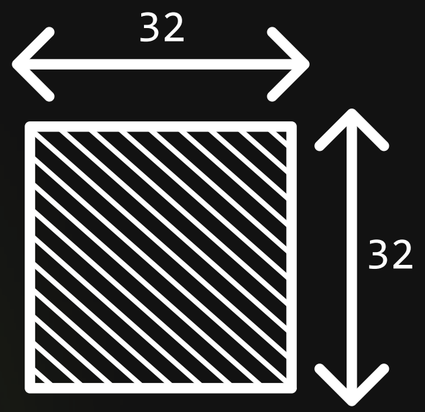
Melakukan hyperparameter tuning pada model CNN kustom untuk menemukan parameter-parameter yang dapat meningkatkan akurasi, presisi, dan recall dari model

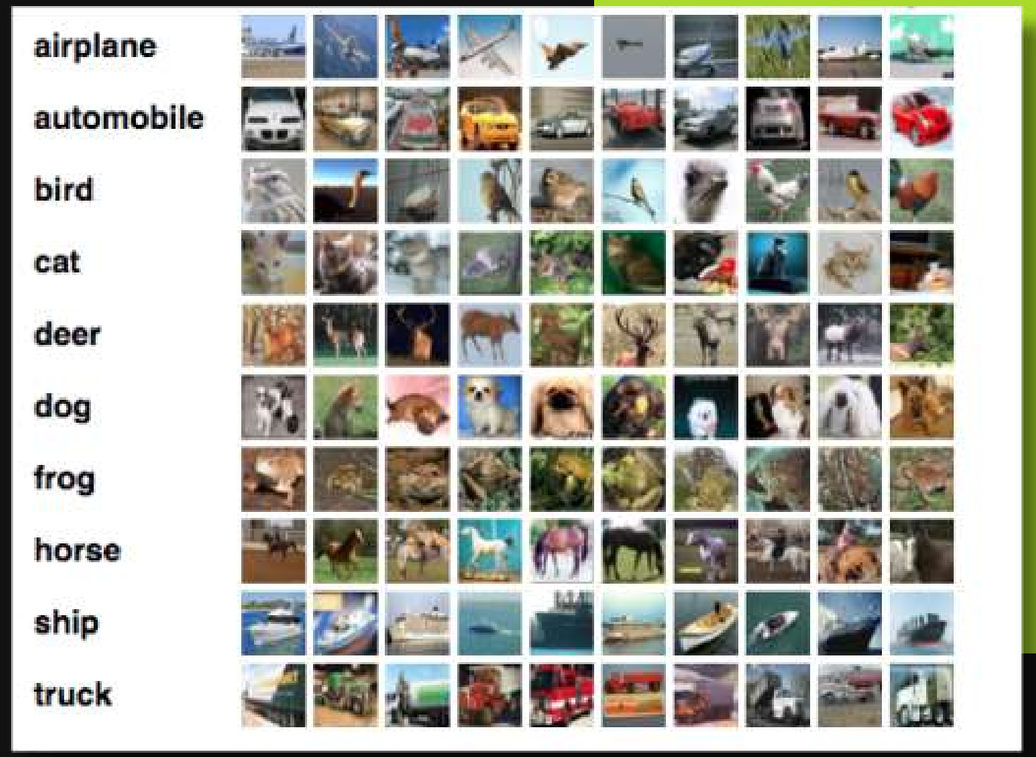
## 03.

### Perbandingan Model

Membandingkan model CNN kustom dengan model Resnet18 pretrained yang sudah disesuaikan untuk dataset CIFAR-10 dari modul PyTorch untuk mengetahui mana model yang dapat mengklasifikasi dataset CIFAR-10 dengan baik

# ABOUT CIFAR-10

# METODOLOGI UMUM & PERSIAPAN DATA

**01**

## Set Seed

`set_seed(42)`
untuk memastikan bahwa setiap kali kode dijalankan, hasil yang diperoleh akan konsisten, terutama dalam hal inisialisasi bobot model, pembagian data, dan operasi lain yang melibatkan proses acak.

**02**

## Slice dataset

Bagi dataset full training menjadi:


**03**

## Transformasi train, eval, & testing

Transformasi dataset training, evaluasi, dan testing dengan:

- mean : 0.5, 0.5, 0.5
- std : 0.5, 0.5, 0.5

**04**

## Data augmentasi untuk training

```
transforms.RandomCrop(32, padding=4)
transforms.RandomHorizontalFlip()
transforms.ToTensor()
```

# CNN KUSTOM

# ARSITEKTUR CNN KUSTOM

```python
class CIFAR10NetWithFeatureMaps(nn.Module):
    def __init__(self, config):
        super(CIFAR10NetWithFeatureMaps, self).__init__()
        conv_filters = config.get('conv_filters', [64, 128, 256])
        kernel_size = config.get('kernel_size', 3)
        pool_type = config.get('pool_type', 'max')
        fc_neurons = config.get('fc_neurons', 1024)
        dropout_rate = config.get('dropout_rate', 0.4)
        padding = kernel_size // 2

        # Layer Konvolusi Pertama
        self.conv1 = nn.Conv2d(3, conv_filters[0], kernel_size=kernel_size, padding=padding)
        self.bn1 = nn.BatchNorm2d(conv_filters[0])

        # Layer Konvolusi Kedua
        self.conv2 = nn.Conv2d(conv_filters[0], conv_filters[1], kernel_size=kernel_size, padding=padding)
        self.bn2 = nn.BatchNorm2d(conv_filters[1])

        # Layer Konvolusi Ketiga
        self.conv3 = nn.Conv2d(conv_filters[1], conv_filters[2], kernel_size=kernel_size, padding=padding)
        self.bn3 = nn.BatchNorm2d(conv_filters[2])

        # Layer Pooling
        if pool_type == 'max':
            self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        elif pool_type == 'avg':
            self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        else: # Jika tipe pooling tidak dikenali
            raise ValueError("pool_type harus 'max' atau 'avg'")

        fc_input_features = conv_filters[2] * 4 * 4
        self.fc1 = nn.Linear(fc_input_features, fc_neurons)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc2 = nn.Linear(fc_neurons, 10)
```

```python
class CIFAR10NetWithFeatureMaps(nn.Module):
    def __init__(self, config):
        super(CIFAR10NetWithFeatureMaps, self).__init__()
        conv_filters = config.get('conv_filters', [64, 128, 256])
        kernel_size = config.get('kernel_size', 3)
        pool_type = config.get('pool_type', 'max')
        fc_neurons = config.get('fc_neurons', 1024)
        dropout_rate = config.get('dropout_rate', 0.4)
        padding = kernel_size // 2

        # Layer Konvolusi Pertama
        self.conv1 = nn.Conv2d(3, conv_filters[0], kernel_size=kernel_size, padding=padding)
        self.bn1 = nn.BatchNorm2d(conv_filters[0])

        # Layer Konvolusi Kedua
        self.conv2 = nn.Conv2d(conv_filters[0], conv_filters[1], kernel_size=kernel_size, padding=padding)
        self.bn2 = nn.BatchNorm2d(conv_filters[1])

        # Layer Konvolusi Ketiga
        self.conv3 = nn.Conv2d(conv_filters[1], conv_filters[2], kernel_size=kernel_size, padding=padding)
        self.bn3 = nn.BatchNorm2d(conv_filters[2])

        # Layer Pooling
        if pool_type == 'max':
            self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        elif pool_type == 'avg':
            self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        else: # Jika tipe pooling tidak dikenali
            raise ValueError("pool_type harus 'max' atau 'avg'")

        fc_input_features = conv_filters[2] * 4 * 4
        self.fc1 = nn.Linear(fc_input_features, fc_neurons)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc2 = nn.Linear(fc_neurons, 10)
```

Ini adalah 'resep' atau 'cetak biru' awal untuk membangun jaringan kita, di mana kita menentukan nama model dan menyiapkan semua 'bahan' utama seperti ukuran filter dan tipe lapisan yang akan digunakan.

Di sini, kita membuat lapisan pertama yang bertugas 'melihat' gambar dan mencari pola-pola dasar seperti tepi atau sudut.

Ini adalah lapisan 'penglihatan' kedua , yang mengambil hasil dari lapisan pertama untuk mencari pola yang lebih rumit.

Selanjutnya, kita siapkan lapisan 'penglihatan' ketiga. Lapisan ini akan menggali lebih dalam lagi untuk menemukan detail dan kombinasi pola yang lebih spesifik dari hasil lapisan kedua.

Bagian ini berfungsi untuk meringkas informasi penting dari hasil 'penglihatan' tadi dan mengurangi ukuran data agar lebih efisien, seperti mengambil intisari.

Ini adalah bagian 'pemikir' dari jaringan. Setelah semua pola dikenali, lapisan ini akan memutuskan gambar itu apa. Dropout membantu agar model tidak terlalu 'menghafal' saat belajar, sehingga lebih pintar.

# ARSITEKTUR CNN KUSTOM

```
    def forward(self, x):
        feature_maps = {}

        # Blok Konvolusi 1
        out_conv1 = F.relu(self.bn1(self.conv1(x)))
        feature_maps['conv1'] = out_conv1
        x_pooled1 = self.pool(out_conv1)

        # Blok Konvolusi 2
        out_conv2 = F.relu(self.bn2(self.conv2(x_pooled1)))
        feature_maps['conv2'] = out_conv2
        x_pooled2 = self.pool(out_conv2)

        # Blok Konvolusi 3
        out_conv3 = F.relu(self.bn3(self.conv3(x_pooled2)))
        feature_maps['conv3'] = out_conv3
        x_pooled3 = self.pool(out_conv3)

        # Flattening dan Fully Connected Layers
        x_flattened = x_pooled3.view(-1, self.fc1.in_features)
        x_fc1 = F.relu(self.fc1(x_flattened))
        x_dropout = self.dropout(x_fc1)
        final_output = self.fc2(x_dropout)
        return final_output, feature_maps
```

```python
    def forward(self, x):
        feature_maps = {}

        # Blok Konvolusi 1
        out_conv1 = F.relu(self.bn1(self.conv1(x)))
        feature_maps['conv1'] = out_conv1
        x_pooled1 = self.pool(out_conv1)

        # Blok Konvolusi 2
        out_conv2 = F.relu(self.bn2(self.conv2(x_pooled1)))
        feature_maps['conv2'] = out_conv2
        x_pooled2 = self.pool(out_conv2)

        # Blok Konvolusi 3
        out_conv3 = F.relu(self.bn3(self.conv3(x_pooled2)))
        feature_maps['conv3'] = out_conv3
        x_pooled3 = self.pool(out_conv3)

        # Flattening dan Fully Connected Layers
        x_flattened = x_pooled3.view(-1, self.fc1.in_features)
        x_fc1 = F.relu(self.fc1(x_flattened))
        x_dropout = self.dropout(x_fc1)
        final_output = self.fc2(x_dropout)
        return final_output, feature_maps
```

Ini adalah **langkah pertama pemrosesan gambar**: gambar dimasukkan ke lapisan 'penglihatan' pertama, hasilnya (fitur) dicatat, lalu diringkas.

**Langkah kedua pemrosesan:** hasil ringkasan dari langkah sebelumnya dimasukkan ke lapisan 'penglihatan' kedua, fitur-fitur baru dicatat, dan diringkas lagi.

**Langkah ketiga pemrosesan:** data yang sudah diproses dua kali kini dimasukkan ke lapisan 'penglihatan' ketiga. Di sini, fitur-fitur gambar akan dikenali lebih detail lagi, hasilnya dicatat, dan kembali diringkas.

**Tahap akhir:** semua informasi penting yang sudah diringkas **disatukan dan 'diratakan'**, lalu dianalisis oleh bagian 'pemikir' untuk menghasilkan tebakan akhir tentang isi gambar.

# ARSITEKTUR CNN KUSTOM

| Lapisan | Input Shape | Output Shape | Parameter Utama | Keterangan |
|---|---|---|---|---|
| conv1 | (3, 32, 32) | (128, 32, 32) | Kernel: 3x3, Filters: 128, Padding: 1 | Lapisan konvolusi awal untuk mengekstraksi fitur dari gambar masukan. |
| batchnorm1 | (128, 32, 32) | (128, 32, 32) | Channels: 128 | Normalisasi batch untuk mempercepat dan menstabilkan pelatihan. |
| pool1 (MaxPooling) | (128, 32, 32) | (128, 16, 16) | Kernel: 2x2, Stride: 2 | Pengurangan resolusi spasial (downsampling). |
| conv2 | (128, 16, 16) | (256, 16, 16) | Kernel: 3x3, Filters: 256, Padding: 1 | Ekstraksi fitur tingkat lanjut. |
| batchnorm2 | (256, 16, 16) | (256, 16, 16) | Channels: 256 | Normalisasi batch lanjutan. |
| pool2 (MaxPooling) | (256, 16, 16) | (256, 8, 8) | Kernel: 2x2, Stride: 2 | Reduksi ukuran fitur spasial. |
| conv3 | (256, 8, 8) | (512, 8, 8) | Kernel: 3x3, Filters: 512, Padding: 1 | Konvolusi dalam untuk fitur yang lebih kompleks. |
| batchnorm3 | (512, 8, 8) | (512, 8, 8) | Channels: 512 | Normalisasi batch untuk fitur beresolusi tinggi. |
| pool3 (MaxPooling) | (512, 8, 8) | (512, 4, 4) | Kernel: 2x2, Stride: 2 | Downsampling terakhir sebelum FC. |
| flatten | (512, 4, 4) | 8192 | - | Menyusun ulang tensor ke bentuk vektor 1D. |
| fc1 (Linear) | 8192 | 512 | Input: 8192, Output: 512 | Fully connected layer untuk ekstraksi representasi. |
| dropout | 512 | 512 | Probabilitas: 0.5 | Regularisasi untuk mengurangi overfitting. |
| fc2 (Linear) | 512 | 10 | Input: 512, Output: 10 | Menghasilkan output logits untuk klasifikasi CIFAR-10 (10 kelas). |

# HYPERPARAMETER TUNING

## Parameter Kunci yang Dieksplorasi

- Learning Rate (LR): Dicoba beberapa nilai berbeda: 0.0005, 0.001, 0.01.
  - Tujuan: Menemukan kecepatan belajar optimal.
- Batch Size: Ukuran batch divariasikan: 64, 128, 256.
  - Tujuan: Keseimbangan antara kecepatan training dan stabilitas gradien.
- Arsitektur & Regularisasi Model:
  - Filter Konvolusi: Set filter seperti (64,128,256), (128,256,512), (32,64,128).
  - Neuron Fully Connected (FC): Jumlah neuron 1024 dan 512.
  - Tipe Pooling: Menggunakan max pooling dan avg pooling.
  - Dropout Rate: Variasi rate 0.3, 0.4, 0.5 untuk mencegah overfitting.
  - Weight Decay (L2): Nilai 0 (tanpa decay), 1e-4, 5e-4.

## Strategi Optimasi & Pelatihan

- Optimizer yang Diuji:
  - Adam: Kombinasi momentum dan adaptasi learning rate.
  - SGD: Dengan momentum = 0.9 untuk akselerasi.
  - RMSprop: Optimizer adaptif lainnya.
  - Eksplorasi ini bertujuan mencari metode optimasi paling efektif.
- Durasi & Penghentian Pelatihan (Epochs & Early Stopping):
  - Jumlah Epoch Maksimum: 75 untuk setiap konfigurasi.
  - Early Stopping (Patience): Pelatihan akan dihentikan jika tidak ada peningkatan signifikan pada metrik validasi setelah 15 epoch.
  - Ini untuk efisiensi dan menghindari overfitting.
- Penyesuaian Learning Rate (Scheduler):
  - Metode: Learning rate di-update (umumnya dikurangi) setiap 25 epoch (scheduler_step).
  - Pendekatan ini berbeda dari ReduceLROnPlateau yang berbasis stagnasi metrik.

# HASIL TERBAIK (CONFIG 2 - SGD)

## Parameter yang di-tuning (dan Konfigurasi Model)

- Learning Rate (lr): [0.01]
- Batch Size (batch_size): [64]
- Jumlah Filter Konvolusi (conv_filters): (128, 256, 512)
- Ukuran Kernel (kernel_size): 3
- Tipe Pooling (pool_type): "max"
- Neuron Fully Connected (fc_neurons): 512
- Dropout Rate (dropout_rate): 0.5
- Weight Decay (weight_decay): 5e-4

## Strategi Optimasi & Pelatihan

- SGD (Stochastic Gradient Descent) dengan momentum.
- Momentum: 0.9

## Scheduler: Step Based Scheduler

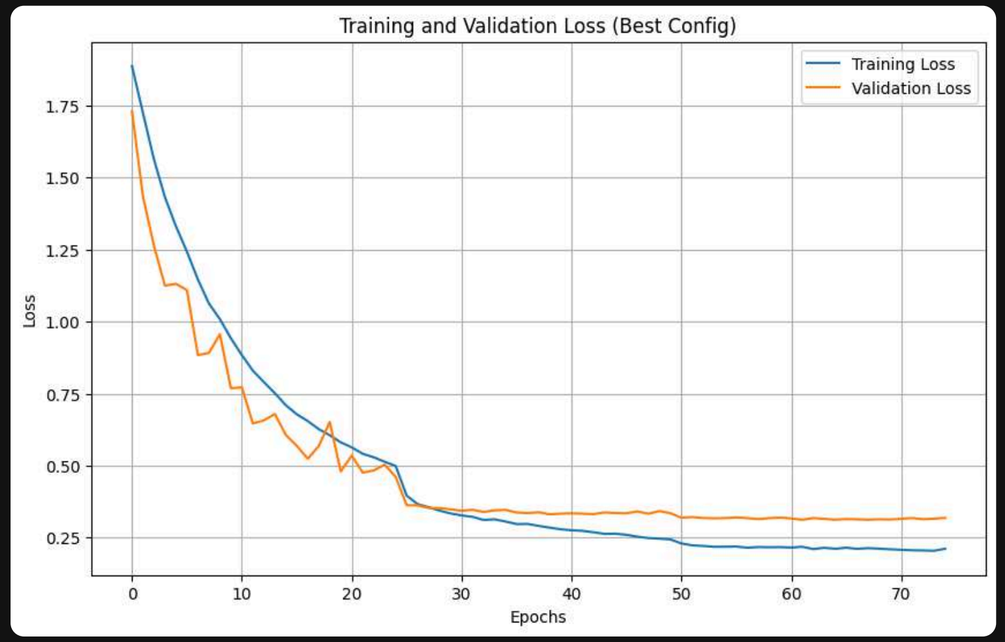
- Scheduler akan menyesuaikan learning rate pada step tertentu.
- Scheduler Step (scheduler_step): 25 epoch (learning rate diubah setiap 25 epoch)

## Epochs: 75, Early Stopping: Patience=15

- 75 epoch dengan early stopping (patience=15) untuk menghentikan pelatihan jika akurasi/loss pada validation set tidak membaik setelah 15 epoch.

## Hasil

- Akurasi training: 93.10%
- Akurasi validasi: 89.46%
- Akurasi tes: 89.26%




# 01.

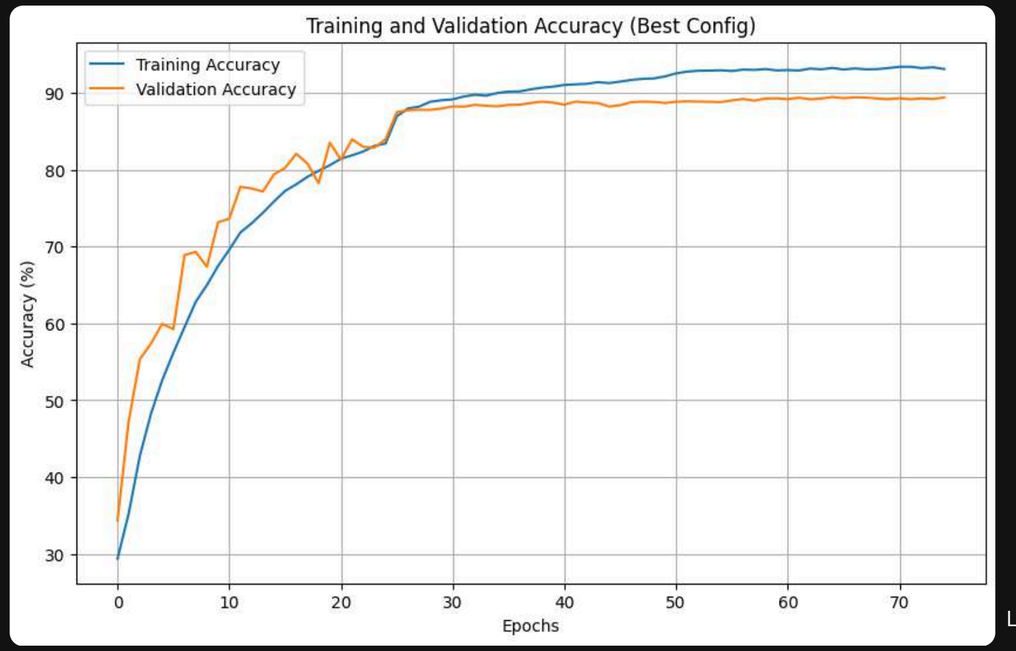
## Kurva Loss

- **Training Loss dan Validation Loss** sama-sama menurun tajam di awal, lalu stabil di sekitar epoch ke-30.
- Validation loss cenderung **datarnya stabil**, tidak naik drastis, artinya **tidak terjadi overfitting.**
- Model belajar dengan baik dan **konvergen secara stabil.**

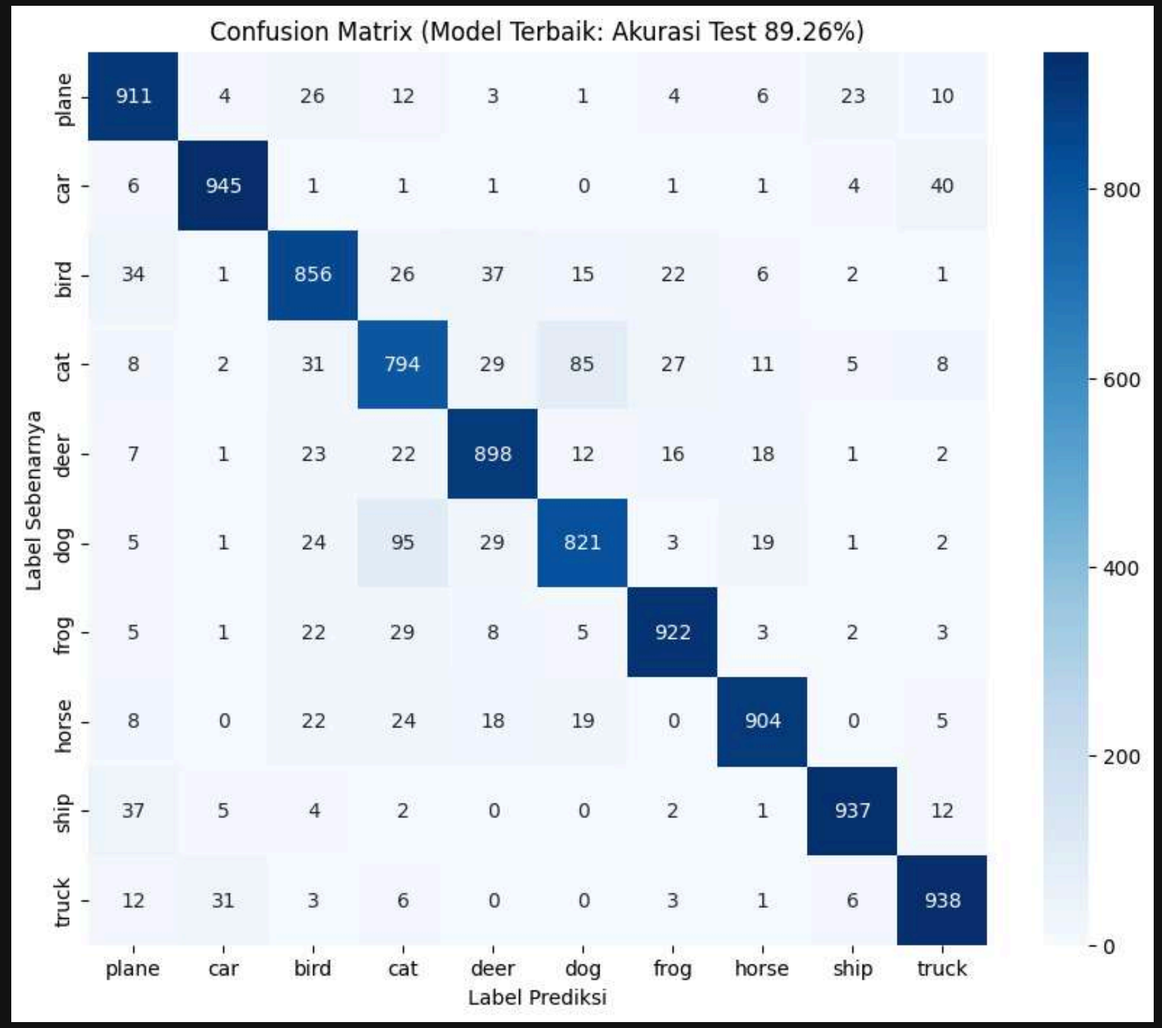
# 02.

## Kurva Akurasi

- **Training Accuracy dan Validation Accuracy** meningkat konsisten, lalu sama-sama stabil setelah sekitar epoch ke-30.
- Tidak ada gap besar antara keduanya, yang menunjukkan model **generalize dengan baik.**
- Akurasi validasi mendekati akurasi training → **model cukup optimal dan tidak overfit.**



# CONFUSION MATRIKS



# METRIK AKURASI

| Kelas | Precision | Recall | f1-score |
|---|---|---|---|
| plane | 0.88 | 0.91 | 0.8948 |
| car | 0.95 | 0.94 | 0.945 |
| bird | 0.85 | 0.86 | 0.855 |
| cat | 0.79 | 0.79 | 0.79 |
| deer | 0.88 | 0.9 | 0.89 |
| dog | 0.86 | 0.82 | 0.8395 |
| frog | 0.92 | 0.92 | 0.92 |
| horse | 0.93 | 0.9 | 0.9148 |
| ship | 0.96 | 0.94 | 0.9499 |
| truck | 0.92 | 0.94 | 0.9299 |


# RESNET 18

Model ResNet18 yang digunakan adalah model pretrained yang ada pada modul PyTorch. Model ResNet18 ini kami sesuaikan agar dapat mengenali dataset CIFAR-10 dengan baik.

# ARSITEKTUR RESNET18

```python
def get_model(num_classes=NUM_CLASSES):

    model = models.resnet18(weights=None)

    model.conv1 = nn.Conv2d(3, 64,
                            kernel_size=(3, 3),
                            stride=(1, 1),
                            padding=(1, 1),
                            bias=False)

    model.maxpool = nn.Identity()

    num_ftrs = model.fc.in_features
    model.fc = nn.Linear(num_ftrs, num_classes)

    return model

model = get_model(NUM_CLASSES)
model = model.to(device)
```

# ARSITEKTUR RESNET18

| Layer (Type:Depth-Idx) | Input Shape | Output Shape | Parameter Utama | Jumlah Parameter | Keterangan |
|---|---|---|---|---|---|
| Conv2d: 1-1 | (3, 32, 32) | (64, 32, 32) | Weight, Bias | 1,728 | Konvolusi awal, kernel 3x3, 64 filter, stride 1, padding 1. |
| BatchNorm2d: 1-2 | (64, 32, 32) | (64, 32, 32) | Gamma, Beta | 128 | Normalisasi batch untuk 64 channel. |
| ReLU: 1-3 | (64, 32, 32) | (64, 32, 32) | - | 0 | Aktivasi ReLU, tidak ada parameter. |
| Identity: 1-4 | (64, 32, 32) | (64, 32, 32) | - | 0 | Placeholder, tidak mengubah data. |
| Sequential: 1-5 | (64, 32, 32) | (64, 32, 32) | - | 0 | Blok residu Layer 1 (2 BasicBlock). |
| ├─BasicBlock: 2-1 | (64, 32, 32) | (64, 32, 32) | Conv2d, BatchNorm2d | 73,984 | 2 Conv2d (36,864) + 2 BatchNorm2d (128), 64 filter, |
| ├─BasicBlock: 2-2 | (64, 32, 32) | (64, 32, 32) | Conv2d, BatchNorm2d | 73,984 | Sama seperti BasicBlock 2-1, tanpa downsampling. |
| Sequential: 1-6 | (64, 32, 32) | (128, 16, 16) | - | 0 | Blok residu Layer 2 (2 BasicBlock). |
| ├─BasicBlock: 2-3 | (64, 32, 32) | (128, 16, 16) | onv2d, BatchNorm2d, Sequentia | 229,888 | 2 Conv2d (73,728 + 147,456), 2 BatchNorm2d (256), |
| ├─BasicBlock: 2-4 | (128, 16, 16) | (128, 16, 16) | Conv2d, BatchNorm2d | 295,424 | 2 Conv2d (147,456), 2 BatchNorm2d (256), tanpa |
| Sequential: 1-7 | (128, 16, 16) | (256, 8, 8) | - | 0 | Blok residu Layer 3 (2 BasicBlock). |
| ├─BasicBlock: 2-5 | (128, 16, 16) | (256, 8, 8) | onv2d, BatchNorm2d, Sequentia | 918,528 | 2 Conv2d (294,912 + 589,824), 2 BatchNorm2d (512), |
| ├─BasicBlock: 2-6 | (256, 8, 8) | (256, 8, 8) | Conv2d, BatchNorm2d | 1,180,672 | 2 Conv2d (589,824), 2 BatchNorm2d (512), tanpa |
| Sequential: 1-8 | (256, 8, 8) | (512, 4, 4) | - | 0 | Blok residu Layer 4 (2 BasicBlock). |
| ├─BasicBlock: 2-7 | (256, 8, 8) | (512, 4, 4) | onv2d, BatchNorm2d, Sequentia | 3,672,064 | 2 Conv2d (1,179,648 + 2,359,296), 2 BatchNorm2d |
| ├─BasicBlock: 2-8 | (512, 4, 4) | (512, 4, 4) | Conv2d, BatchNorm2d | 4,720,640 | 2 Conv2d (2,359,296), 2 BatchNorm2d (1,024), tanpa |
| AdaptiveAvgPool2d: 1- | (512, 4, 4) | (512, 1, 1) | - | 0 | Global average pooling, mengurangi dimensi spasial ke |
| Linear: 1-10 | -512 | -10 | Weight, Bias | 5,130 | Fully connected layer, 512 fitur ke 10 kelas (512x10 + 10 |
| Total | - | - | - | 11,173,962 | Total parameter model ResNet-18. |

# HYPER PARAMETER

**Parameter**

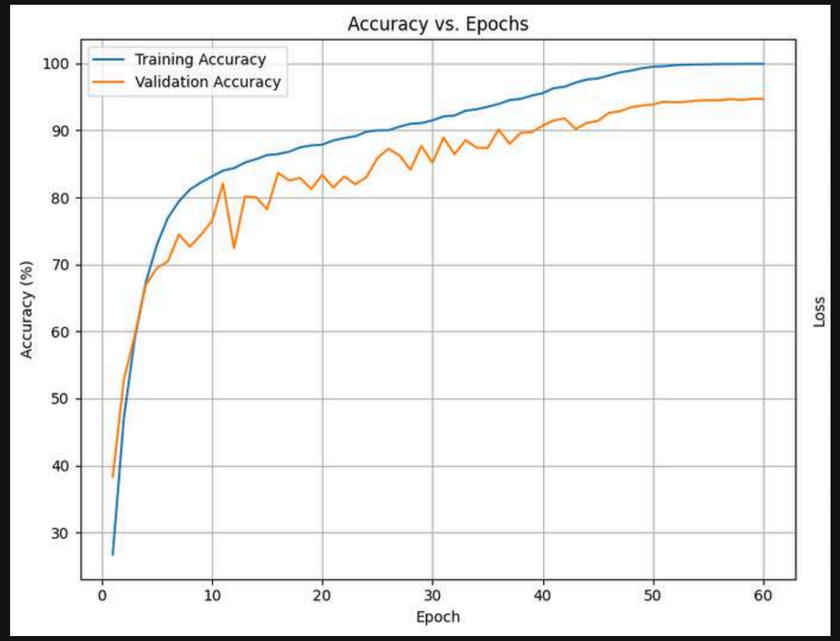
BATCH_SIZE = 128
EPOCHS = 60
NUM_CLASSES = 10 # CIFAR-10 memiliki 10 kelas
LEARNING_RATE = 0.1 # Learning rate awal untuk SGD
MOMENTUM = 0.9 # Momentum untuk SGD
WEIGHT_DECAY = 5e-4 # L2 regularization

**Optimizer**

SGD (Stochastic Gradient Descent) dengan learning rate 0.1, momentum 0.9 (membantu optimizer melewati local minima dan mempercepat konvergensi) dan weight decay 0.0005 (untuk regularisasi L2 untuk mencegah overfitting)

**Scheduler**

Menggunakan CosineAnnealingLR yang dimana secara bertahap mengurangi learning rate mengikuti kurva kosinus. Selain itu, untuk T_max adalah banyaknya epoch, yaitu 60 epoch. Model ini juga menggunakan earliy stopping dengan patience 15




## 02

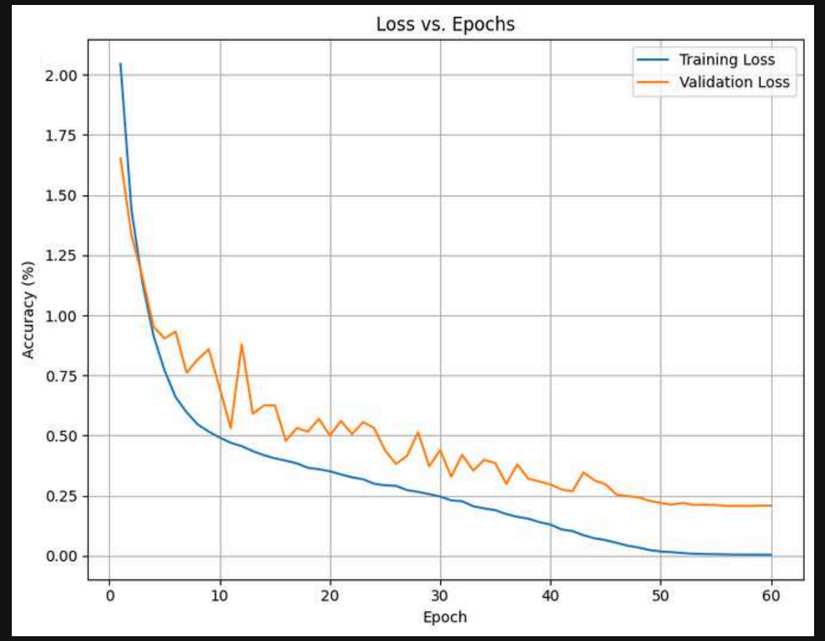
### Kurva Loss

- Training Loss terbaik (pada model terbaik): 0.0054
- Validation Loss terbaik (model disimpan): 0.2069
- Fluktuasi pada kurva validasi, terutama pada loss, mungkin disebabkan oleh variasi dalam batch data validasi

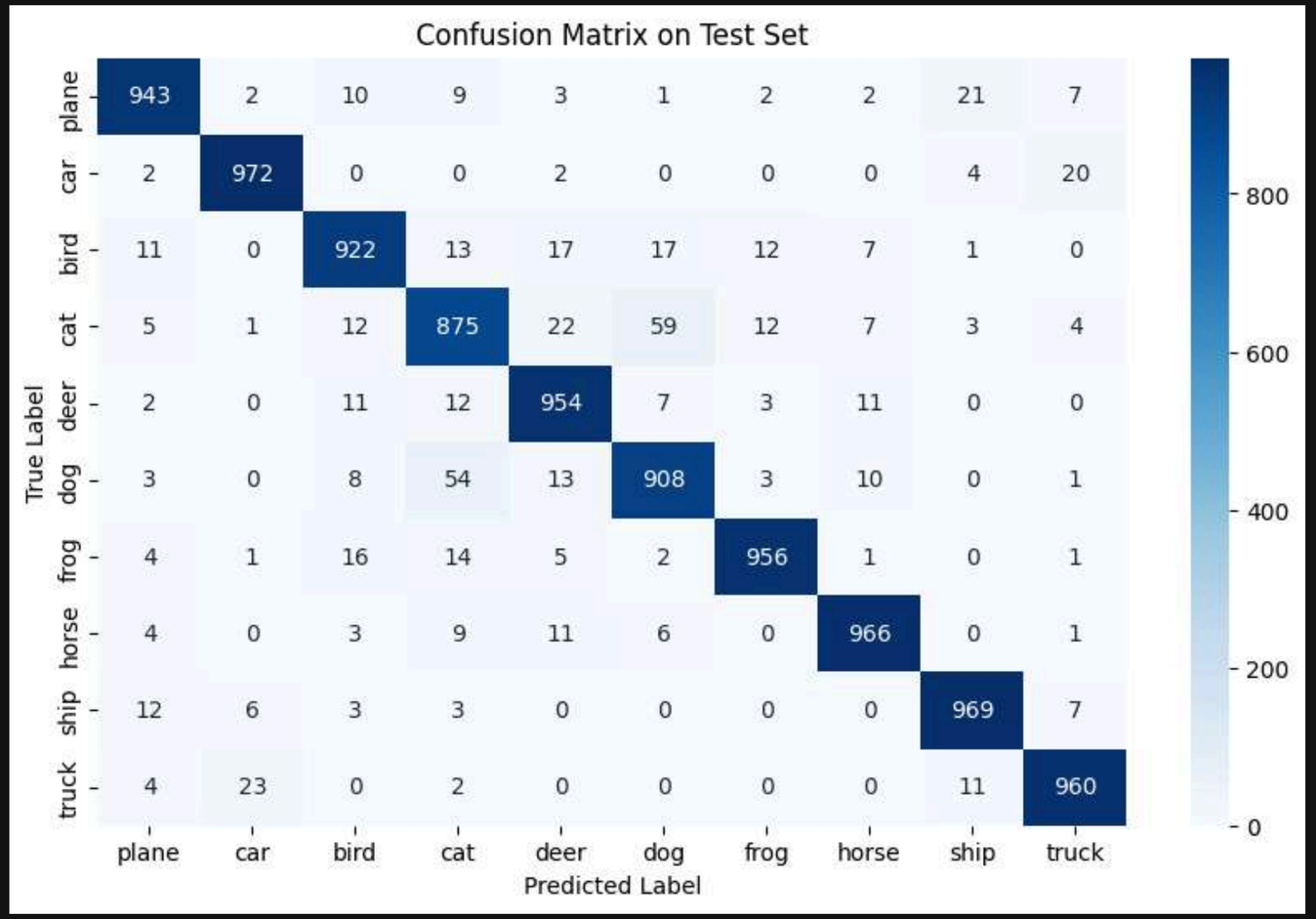
## 01.

### Kurva Akurasi

- Training Akurasi terbaik (pada model terbaik): 99.91%
- Validation Akurasi (pada model terbaik): 94.50%
- Ada indikasi terjadinya sedikit overfitting
- Model ResNet18 yang dimodifikasi ini belajar dengan baik untuk mengklasifikasikan dataset CIFAR-10



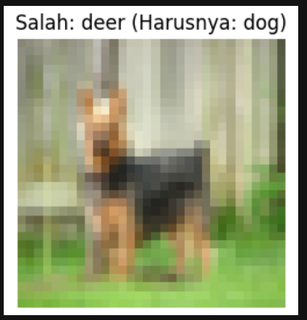
# CONFUSION MATRIKS



# METRIK AKURASI

| Kelas | Precision | Recall | f1-score |
|---|---|---|---|
| plane | 0.9525 | 0.943 | 0.9477 |
| car | 0.9672 | 0.972 | 0.9696 |
| bird | 0.936 | 0.922 | 0.929 |
| cat | 0.8829 | 0.875 | 0.879 |
| deer | 0.9289 | 0.954 | 0.9413 |
| dog | 0.908 | 0.908 | 0.908 |
| frog | 0.9676 | 0.956 | 0.9618 |
| horse | 0.9622 | 0.966 | 0.9641 |
| ship | 0.9604 | 0.969 | 0.9647 |
| truck | 0.959 | 0.96 | 0.9595 |

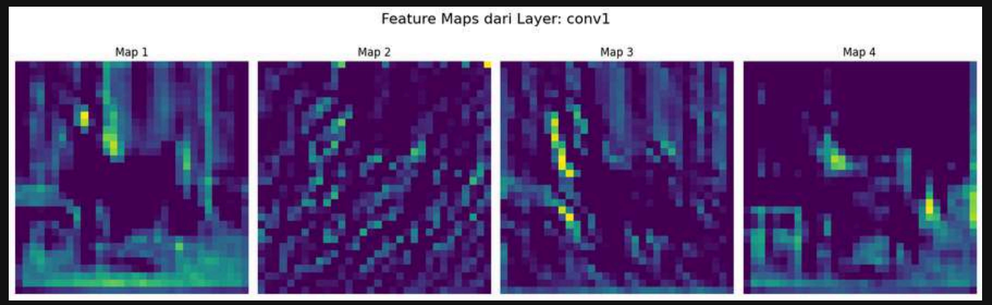
# FEATURE MAPS

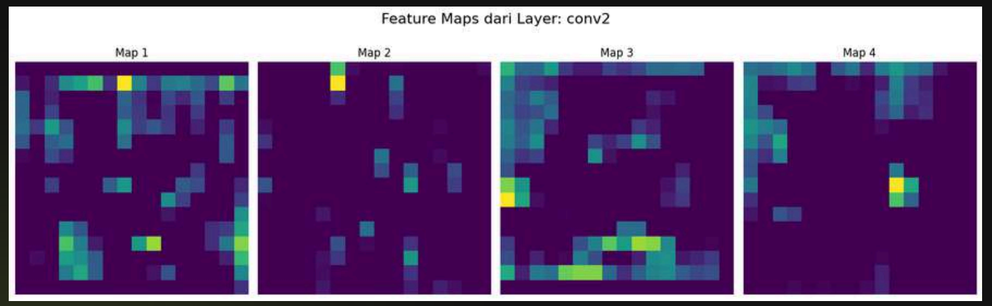


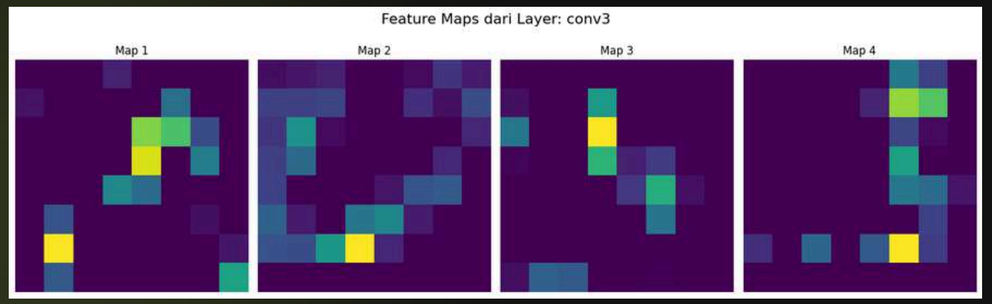

# CNN

Gambar salah klasifikasi:

- Label Sebenarnya: dog
- Prediksi Model : deer




# ANALISIS KESALAHAN KLASIFIKASI CNN

Model CNN salah mengklasifikasikan gambar anjing (dog) sebagai rusa (deer). Hal ini dianalisis melalui visualisasi feature maps dari tiga lapisan konvolusi: conv1, conv2, dan conv3.

01 **Gambar Input** menunjukkan seekor anjing, namun model gagal mengenali ciri khasnya.

02 **Lapisan conv1** mengekstraksi fitur dasar seperti tepi, tekstur, dan kontur. Aktivasi cukup jelas menunjukkan bentuk tubuh hewan.

03 **Lapisan conv2** mulai mereduksi detail dan fokus pada area-area yang lebih spesifik. Aktivasi terlihat menyebar, menunjukkan kurangnya fokus pada bagian penting seperti kepala atau moncong anjing.

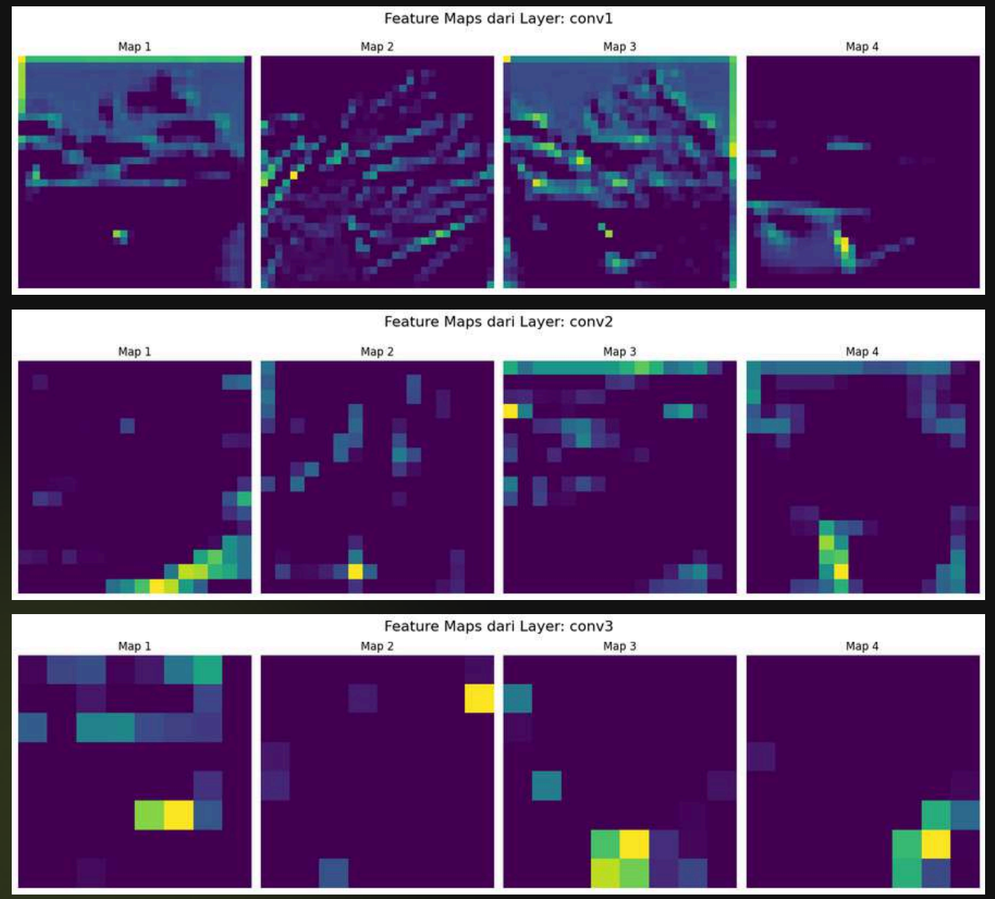
04 **Lapisan conv3** menghasilkan representasi spasial yang sangat sederhana. Aktivasi hanya muncul di area terbatas, menunjukkan bahwa informasi penting mulai hilang atau tidak cukup kuat untuk membedakan dengan kelas deer.

**Kesimpulan**:
Model gagal menangkap fitur diskrit antara dog dan deer, kemungkinan karena kesamaan visual (warna, posisi, bentuk tubuh), keterbatasan data pelatihan, atau arsitektur CNN yang belum optimal dalam menekankan fitur khas kelas dog di lapisan yang lebih dalam.


# CNN

Gambar benar klasifikasi:

- Label Sebenarnya: ship
- Prediksi Model : ship




# ANALISIS KLASIFIKASI CNN

Gambar menunjukkan hasil aktivasi feature maps dari tiga lapisan konvolusi (conv1, conv2, conv3) pada model CNN untuk citra dengan label asli "ship" dan prediksi model juga "ship".

**01**

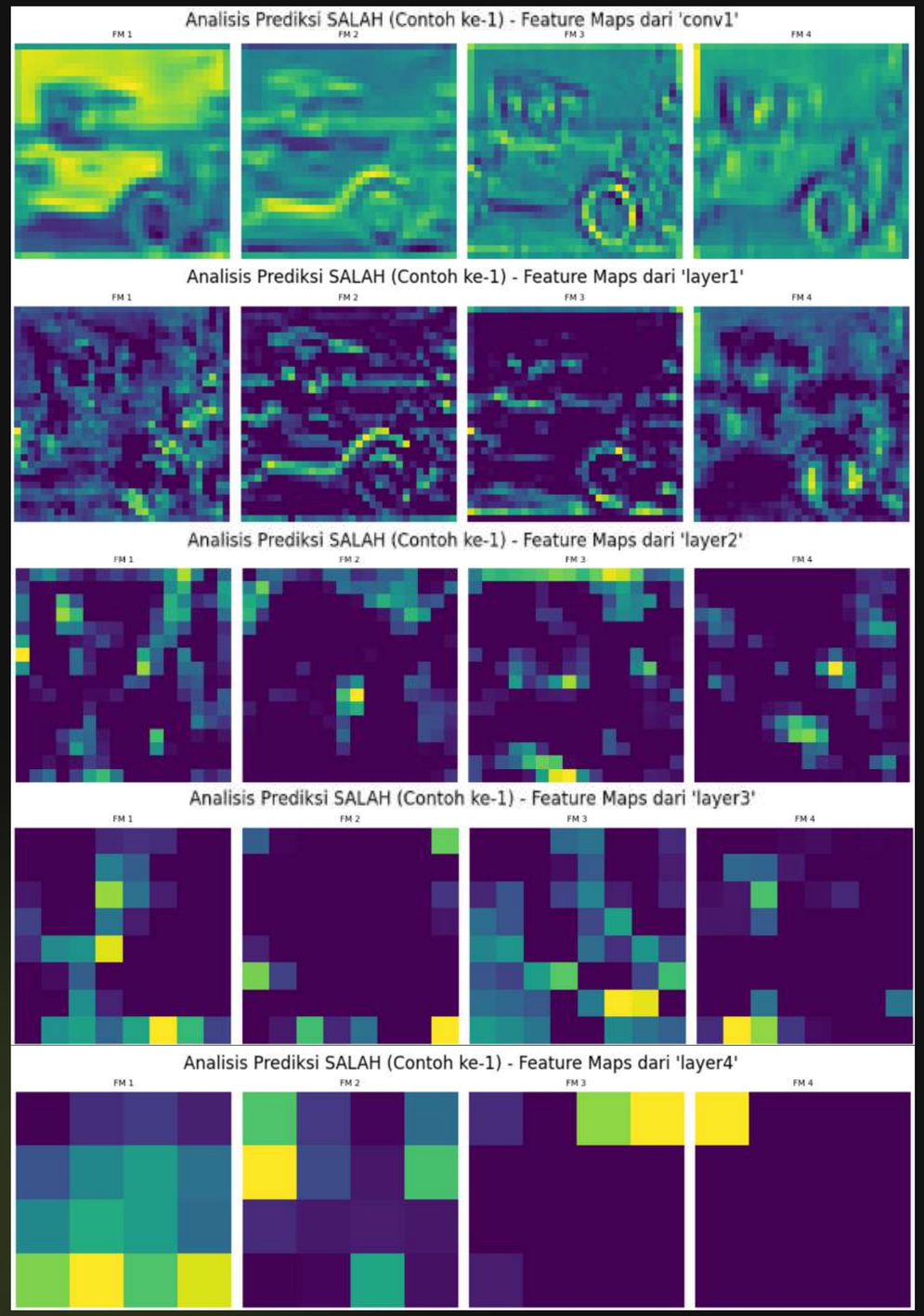
L**apisan conv1** menampilkan feature maps dengan detail tinggi seperti tepi dan pola tekstur dari citra asli. Hal ini umum karena lapisan awal CNN menangkap fitur lokal yang bersifat rendah tingkat (low-level features).

**02**

**Lapisan conv2** menunjukkan fitur yang lebih sederhana, dengan representasi spasial yang semakin kabur. Ini menunjukkan bahwa model mulai menyaring informasi yang lebih penting untuk klasifikasi.

**03**

**Lapisan conv3** menghasilkan aktivasi yang sangat spars (jarang), artinya hanya bagian tertentu dari gambar yang dianggap relevan oleh model pada tahap akhir ekstraksi fitur. Ini mencerminkan fokus model pada fitur high-level yang esensial untuk pengambilan keputusan klasifikasi.




# RESNET18

Gambar salah klasifikasi:

- Label Sebenarnya: car
- Prediksi Model : truck

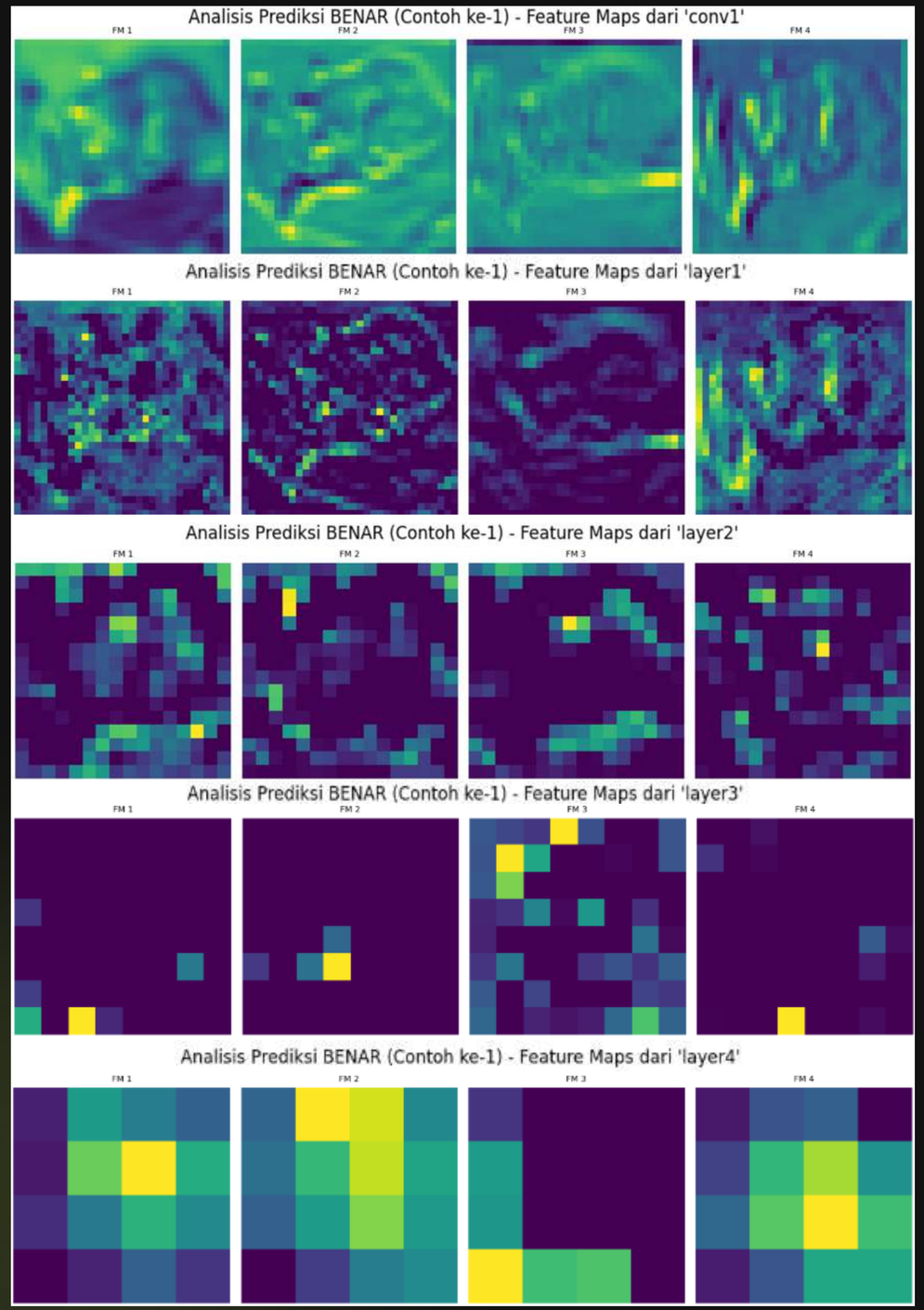
## ANALISIS KESALAHAN KLASIFIKASI RESNET18

Model ResNet18 salah mengklasifikasikan gambar yang sebenarnya berlabel "car" menjadi "truck".

**01** **Layer Awal (conv1 & layer1)** menampilkan banyak detail tekstur dan tepi dari gambar asli. Ini menunjukkan bahwa layer awal menangkap fitur dasar seperti kontur, bentuk, dan tekstur.

**02** **Layer Tengah (layer2 & layer3)** banyak bagian gambar kehilangan detail asli dan informasi spasial mulai disaring. Ini berarti model mulai memfokuskan diri pada fitur yang lebih konseptual, bukan visual mentah.

**03** **Layer Akhir (layer4)** aktivasi menjadi sangat kasar dan terbatas pada area tertentu saja. Ini menunjukkan bahwa model menekankan fitur-fitur yang dianggap penting untuk klasifikasi akhir, meskipun dalam kasus ini fitur yang dianggap penting tersebut justru menyebabkan kesalahan klasifikasi.




# RESNET18

Gambar salah klasifikasi:

- Label Sebenarnya: cat
- Prediksi Model : cat

## ANALISIS KLASIFIKASI RESNET18

Model ResNet18 benar mengklasifikasikan gambar pada dataset CIFAR-10, khususnya untuk kelas “cat”.

**01** **Layer Awal (conv1 & layer1)** banyak detail tekstur dan bentuk seperti bulu dan kontur tubuh kucing masih terlihat jelas. Ini menunjukkan bahwa layer awal fokus pada fitur dasar seperti tepi, warna, dan bentuk.

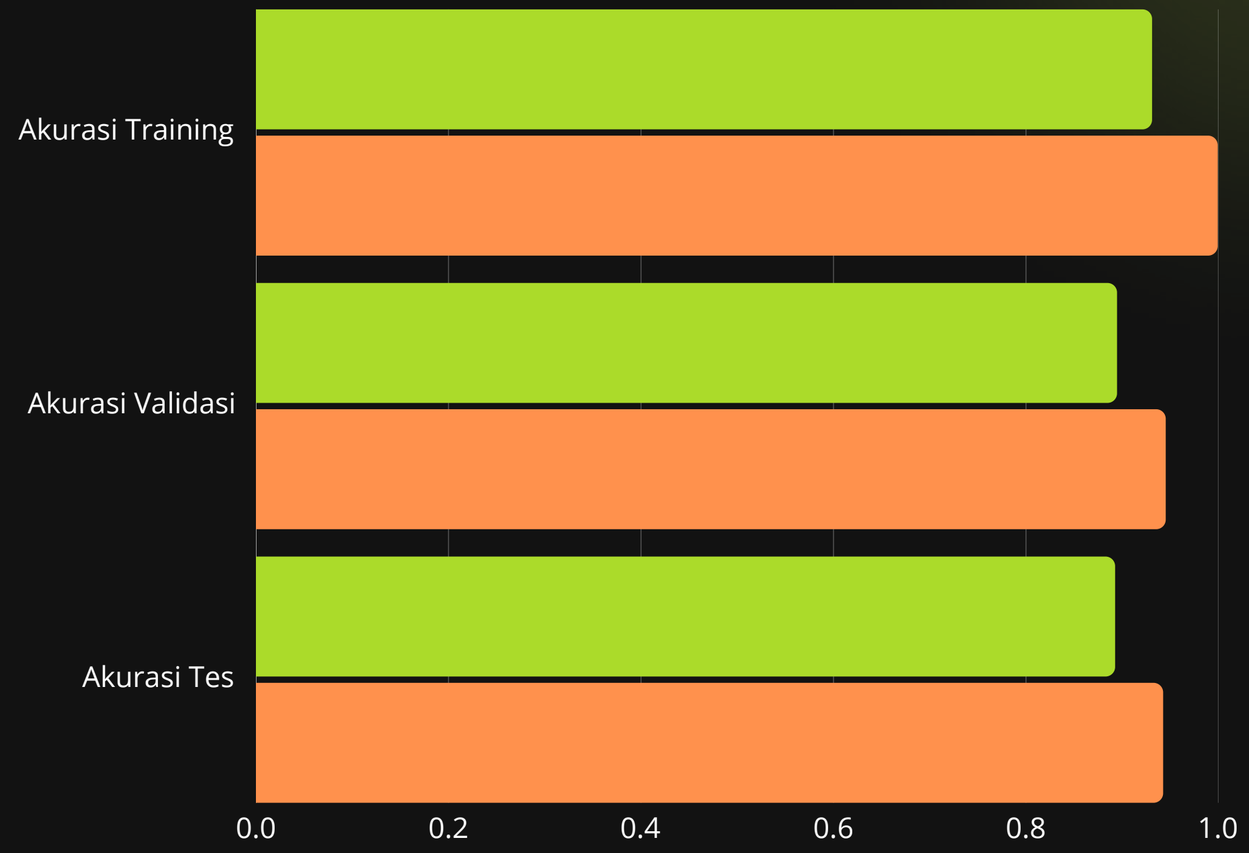
**02** **Layer Tengah (layer2 & layer3)** aktivasi mulai mengabstraksi informasi visual menjadi representasi spasial dan pola, yang umum dilakukan CNN untuk menangkap pola-pola penting antar kelas.

**03** **Layer Akhir (layer4) a**ktivasi lebih fokus pada area tertentu, yang menunjukkan bahwa model berhasil menangkap fitur diskrit yang khas dari kucing. Pola-pola ini digunakan oleh model untuk menghasilkan keputusan klasifikasi akhir.

# CNN KUSTOM VS RESNET18

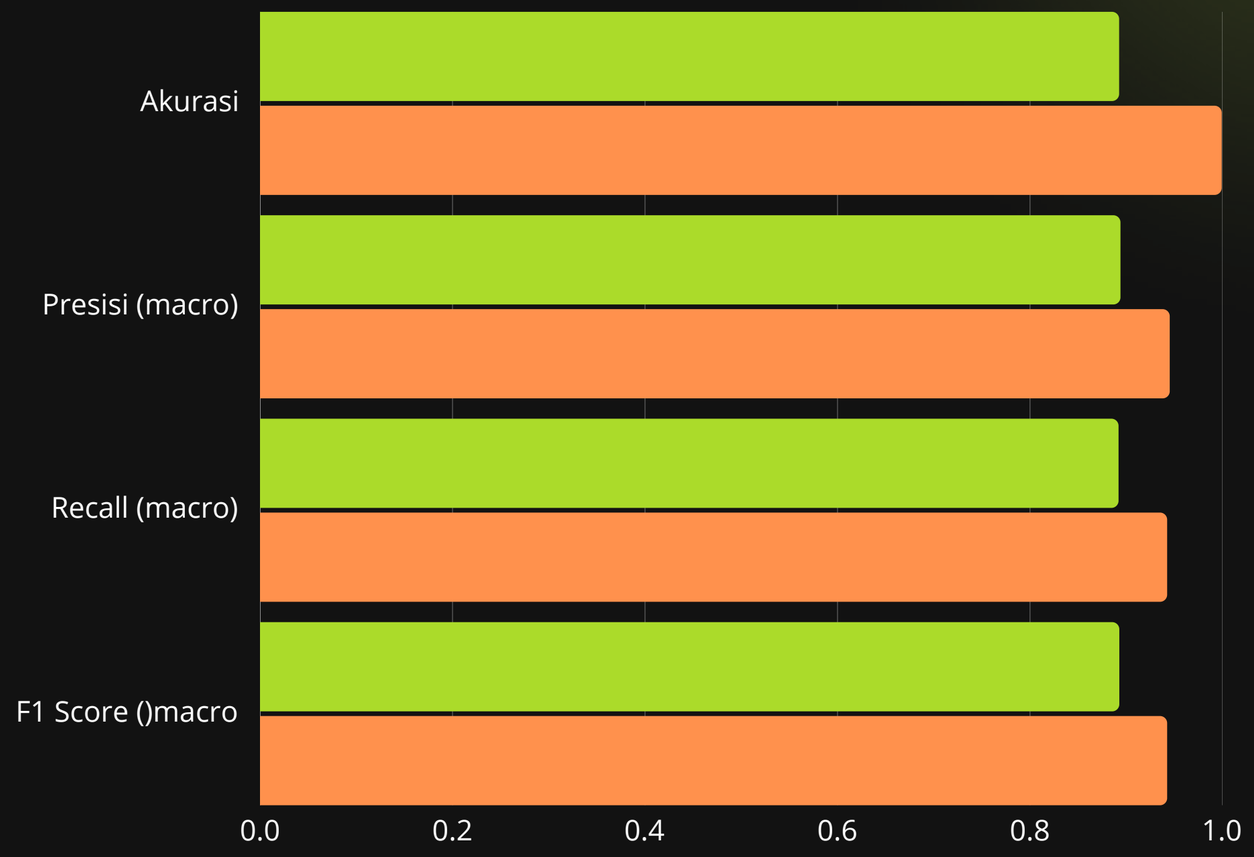
# AKURASI

| CNN Kustom | ResNet18 |
| --- | --- |
| **93.10%** *Akurasi training* | **99.92%** *Akurasi training* |
| **89.46%** *Akurasi validasi* | **94.52%** *Akurasi validasi* |
| **89.26%** *Akurasi tes* | **94.25%** *Akurasi tes* |



# KINERJA MODEL

| CNN Kustom | ResNet18 |
|---|---|
| **89.26%** *Akurasi* | **99.92%** *Akurasi training* |
| **89.4%** *Presisi (macroaverage)* | **94.52%** *Akurasi validasi* |
| **89.2%** *Recall (macroaverage)* | **94.25%** *Akurasi tes* |
| **89.28%** *F1 score (macroaverage)* | **94.25%** *F1 score (macroaverage)* |



# AKURASI PER KELAS

| CNN | | RESNET 18 |
|---|---|---|
| 91.10% | Plane | 94.30% |
| 94.50% | car | 97.20% |
| 85.60% | bird | 92.20% |
| 79.40% | cat | 87.50% |
| 89.80% | deer | 95.40% |
| 82.10% | dog | 90.80% |
| 92.20% | frog | 95.60% |
| 90.40% | horse | 96.60% |
| 93.70% | ship | 96.90% |
| 93.80% | truck | 96.00% |

# LOG TRAINING MODEL

| CNN | | RESNET 18 |
|---|---|---|
| 93.10% | Akurasi Training | 99.92% |
| 89.40% | Akurasi Validasi | 94.52% |
| 0.212 | Train Loss | 0.0051 |
| 0.319 | Validasi Loss | 0.2157 |
| Tidak Overfit | Overfit | Ada indikasi overfit |
| Stabil | Stabilitas | Agak fluktuatif |
| Model cukup kuat & generalisasi baik | Kesimpulan | Model kuat & generalisasi sangat baik |

# SIMPULAN

# CNN VS RESNET18

| CNN | | RESNET18 |
|---|---|---|
| 89.26% | Akurasi test | 94.25% |
| Sedang (3 lapis konvolusi) | Kedalaman jaringan | Dalam (1 konvolusi di dalamnya 4 layer |
| Baik (akurasi tes konsisten) | Kemampuan generalisasi | Sangat baik |
| Tidak Overfitting | kinerja generalisasi model | Ada indikasi overfitting |
| Deteksi fitur dasar | Analisis feature map | Visualisasi Ekstraksi Fitur |
| SGD, LR 0.01, batch 64 | Optimasi & Pelatihan | SGD, LR 0.01, batch 128 |
| Custom CNN, 3 blok, FC | Arsitektur | ResNet18 khusus CIFAR-10 |

## Kinerja Akurasi

- **CNN Kustom**:
  - Akurasi Tes: 89.26%
  - F1 Score: 89.28%
  - Tidak overfitting, performa stabil
- **ResNet18 (pretrained dan disesuaikan):**
  - Akurasi Tes: 94.25%
  - F1 Score: 94.25%
  - Kinerja lebih tinggi, namun terdapat indikasi overfitting ringan.

## Kompleksitas Arsitektur

- **CNN Kustom** terdiri dari 3 lapis konvolusi sederhana.
- **ResNet18** menggunakan arsitektur dalam dan residual blocks, memungkinkan pembelajaran fitur kompleks secara lebih efektif.

## Generalisasi dan Ekstraksi Fitur

- **CNN Kustom** memiliki kemampuan generalisasi yang baik tanpa overfit, tetapi kurang kuat dalam membedakan fitur mirip antar kelas.
- **ResNet18** menunjukkan kemampuan ekstraksi fitur tingkat tinggi, terutama pada layer akhir, namun rentan overfitting jika tidak disesuaikan secara optimal.

## Visualisasi Feature Maps

- Visualisasi menunjukkan **ResNet18** lebih fokus pada fitur penting, namun kadang salah prediksi karena fokus pada bagian gambar yang keliru.
- **CNN Kustom** cenderung kehilangan informasi penting di layer akhir ketika terjadi salah klasifikasi.

# THANK YOU

MAKASI MAS