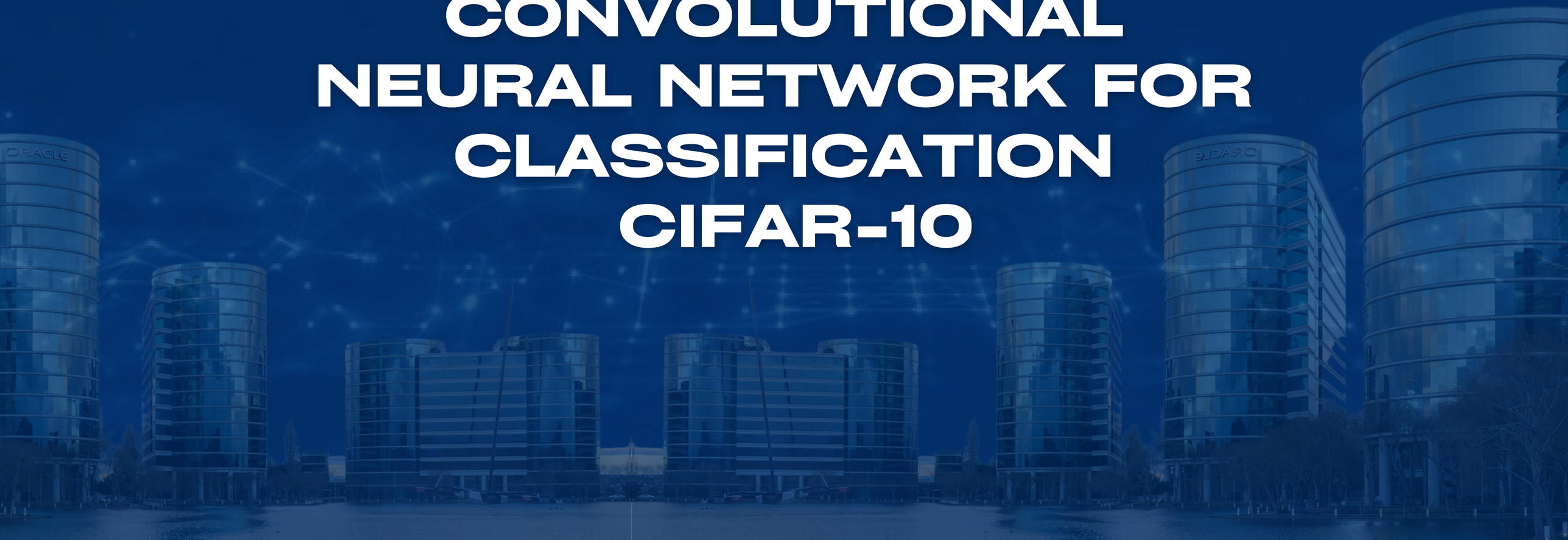
June 5, 2025

CONVOLUTIONAL NEURAL NETWORK FOR CLASSIFICATION CIFAR-10



TUJUAN

Mengklasifikasikan 10 kelas pada dataset CIFAR-10 menggunakan CNN

Melakukan hyperparameter tuning untuk meningkatkan akurasi

Membandingkan performa CustomCNN dengan arsitektur lain (ResNet)

Menganalisis feature maps dan kesalahan prediksi melalui confusion matrix

DATASET

- **Dataset CIFAR-10 terdiri dari 60.000 gambar berwarna (32x32 piksel).**
- **10 kelas: plane, car, bird, cat, deer, dog, frog, horse, ship, truck.**
- **Pembagian data:**
  - ▶ Training: 45.000 gambar (90% dari data train).
  - ▶ Validasi: 5.000 gambar (10% dari data train).
  - ▶ Test: 10.000 gambar.
- **set_seed(42) digunakan untuk reproducibility**

AUGMENTASI DATA

Training Set (transform_train):

▶ RandomCrop(32, padding=4): Memotong gambar secara acak
dengan padding.
▶ RandomHorizontalFlip(): Membalik gambar secara horizontal
secara acak.
▶ ToTensor(): Mengubah gambar ke tensor PyTorch.
▶ Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)): Normalisasi ke
[-1, 1].

Validasi dan Test Set (transform_eval):

▶ ToTensor(): Mengubah gambar ke tensor.
▶ Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)): Normalisasi ke
[-1, 1].


CNN

ARSITEKTUR

```python
class CustomCNN(nn.Module):  # Definisi arsitektur CustomCNN sebagai turunan dari nn.Module
    def __init__(self):  # Inisialisasi model dengan lapisan konvolusi, pooling, fully connected, dan normalisasi
        super(CustomCNN, self).__init__()  # Memanggil konstruktor kelas induk nn.Module
        self.conv1 = nn.Conv2d(3, 64, 3, padding=1)  # Lapisan konvolusi pertama: input 3 channel (RGB), output 64 filter, kernel 3x3, padding 1
        self.conv2 = nn.Conv2d(64, 128, 3, padding=1)  # Lapisan konvolusi kedua: input 64 channel, output 128 filter, kernel 3x3, padding 1
        self.conv3 = nn.Conv2d(128, 256, 3, padding=1)  # Lapisan konvolusi ketiga: input 128 channel, output 256 filter, kernel 3x3, padding 1
        self.pool = nn.MaxPool2d(2, 2)  # Lapisan max pooling: kernel 2x2, stride 2
        self.fc1 = nn.Linear(256 * 4 * 4, 1024)  # Lapisan fully connected pertama: input 256*4*4, output 1024 neuron
        self.fc2 = nn.Linear(1024, 10)  # Lapisan fully connected kedua: input 1024 neuron, output 10 kelas (CIFAR-10)
        self.dropout = nn.Dropout(0.4)  # Lapisan dropout dengan probabilitas 0.4 untuk mencegah overfitting
        self.batchnorm1 = nn.BatchNorm2d(64)  # Batch normalization untuk lapisan konvolusi pertama (64 channel)
        self.batchnorm2 = nn.BatchNorm2d(128)  # Batch normalization untuk lapisan konvolusi kedua (128 channel)
        self.batchnorm3 = nn.BatchNorm2d(256)  # Batch normalization untuk lapisan konvolusi ketiga (256 channel)

    def forward(self, x):  # Definisi alur data melalui jaringan (forward pass)
        x = self.pool(F.relu(self.batchnorm1(self.conv1(x))))  # Konvolusi 1 -> BatchNorm -> ReLU -> MaxPooling
        x = self.pool(F.relu(self.batchnorm2(self.conv2(x))))  # Konvolusi 2 -> BatchNorm -> ReLU -> MaxPooling
        x = self.pool(F.relu(self.batchnorm3(self.conv3(x))))  # Konvolusi 3 -> BatchNorm -> ReLU -> MaxPooling
        x = x.view(-1, 256 * 4 * 4)  # Meratakan output untuk input ke lapisan fully connected
        x = self.dropout(F.relu(self.fc1(x)))  # Fully connected 1 -> ReLU -> Dropout
        x = self.fc2(x)  # Fully connected 2 (output logits untuk 10 kelas)
        return x  # Mengembalikan output jaringan
```

ARSITEKTUR

Lapisan	Input Shape	Output Shape	Parameter Utama	Keterangan
conv1	(3, 32, 32)	(64, 32, 32)	Kernel: 3x3, Filters: 64, Padding: 1	Konvolusi pertama untuk ekstraksi fitur
batchnorm1	(64, 32, 32)	(64, 32, 32)	Channels: 64	Normalisasi untuk stabilisasi pelatihan
ReLU	(64, 32, 32)	(64, 32, 32)	-	Aktivasi non-linear
pool (MaxPooling)	(64, 32, 32)	(64, 16, 16)	Kernel: 2x2, Stride: 2	Reduksi dimensi spasial
conv2	(64, 16, 16)	(128, 16, 16)	Kernel: 3x3, Filters: 128, Padding: 1	Konvolusi kedua untuk fitur lebih kompleks
batchnorm2	(128, 16, 16)	(128, 16, 16)	Channels: 128	Normalisasi untuk stabilisasi pelatihan
ReLU	(128, 16, 16)	(128, 16, 16)	-	Aktivasi non-linear
pool (MaxPooling)	(128, 16, 16)	(128, 8, 8)	Kernel: 2x2, Stride: 2	Reduksi dimensi spasial
conv3	(128, 8, 8)	(256, 8, 8)	Kernel: 3x3, Filters: 256, Padding: 1	Konvolusi ketiga untuk fitur lebih kompleks
batchnorm3	(256, 8, 8)	(256, 8, 8)	Channels: 256	Normalisasi untuk stabilisasi pelatihan
ReLU	(256, 8, 8)	(256, 8, 8)	-	Aktivasi non-linear
pool (MaxPooling)	(256, 8, 8)	(256, 4, 4)	Kernel: 2x2, Stride: 2	Reduksi dimensi spasial
view (Flatten)	(256, 4, 4)	4096	-	Meratakan output untuk lapisan FC
fc1	4096	1024	Input: 25644 (4096), Output: 1024	Lapisan FC untuk transformasi fitur
ReLU	1024	1024	-	Aktivasi non-linear
dropout	1024	1024	Probability: 0.4	Mencegah overfitting
fc2	1024	10	Input: 1024, Output: 10	Output logits untuk 10 kelas (CIFAR-10)

HYPERPARAMETER TUNING

Parameter yang di-tuning:

- Learning Rate (lr): [0.001, 0.003]
- Batch Size (batch_size): [32, 64, 128]

Optimizer: Adam

Adam, yang menggabungkan momentum dan adaptasi learning rate.

Scheduler: ReduceLROnPlateau (mode='max', factor=0.1, patience=3)

mengurangi lr sebesar 0.1 jika akurasi validasi tidak meningkat setelah 3 epoch.

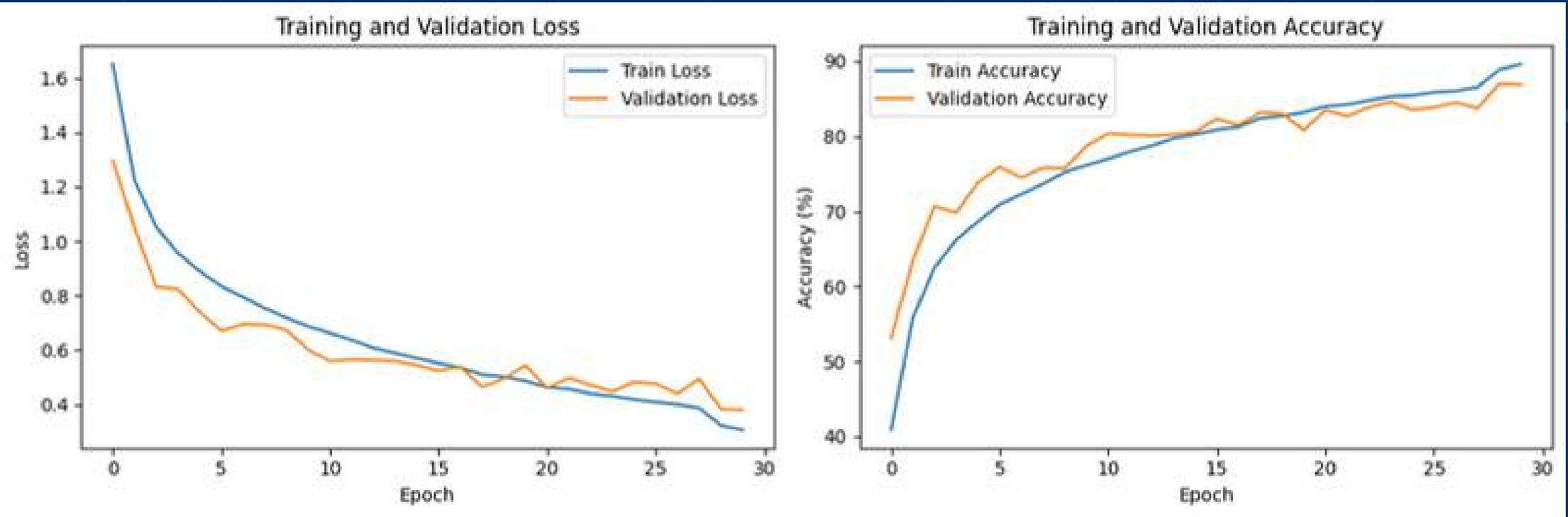
Epochs: 30, Early Stopping: Patience=15

30 epoch dengan early stopping (patience=15) untuk menghentikan pelatihan jika akurasi stagnan.

Hasil:

Best lr: 0.001, Best batch_size: 32 (Val Acc: 87%).

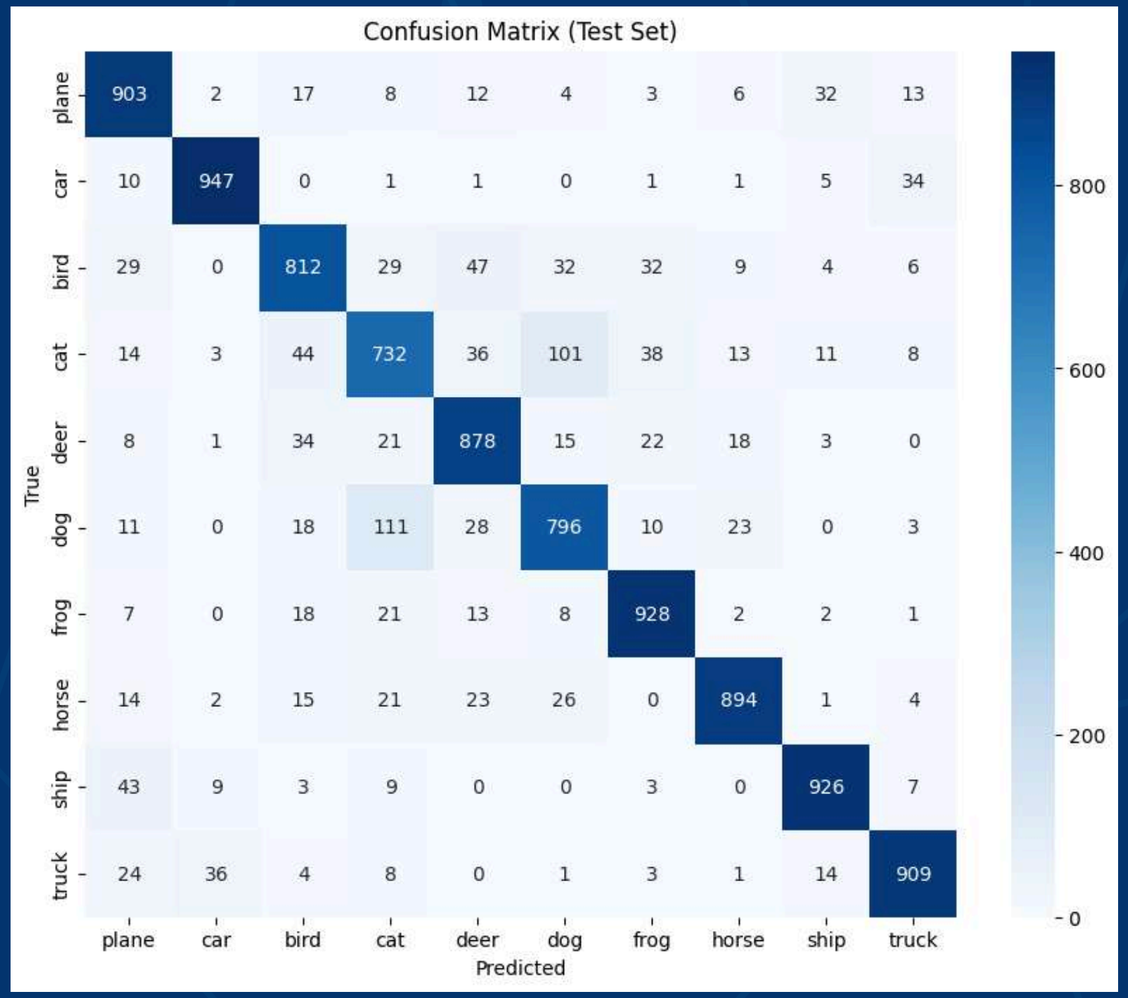
PLOTTING LOSS & AKURASI



PLOTTING LOSS & AKURASI

1. **Grafik Training and Validation Loss:**
   - Train Loss dan Validation Loss sama-sama menurun seiring bertambah epoch.
   - Menunjukkan bahwa model berhasil belajar dari data dan tidak mengalami overfitting.
   - Loss validasi sempat stagnan namun kembali menurun, yang berarti model tetap generalisasi dengan baik.
   - **Kesimpulan:** Model menunjukkan performa pelatihan yang stabil dan efektif, dengan tren loss yang konsisten menurun.
2. **Grafik Training and Validation Accuracy:**
   - Akurasi pelatihan meningkat secara konsisten hingga mendekati 90%.
   - Akurasi validasi juga meningkat dan cenderung mendekati akurasi pelatihan, meskipun sedikit fluktuatif.
   - Tidak ada gap besar antara akurasi pelatihan dan validasi, yang menandakan model tidak overfitting
   - **Kesimpulan:** Model memiliki kinerja yang baik, mampu mengklasifikasikan data dengan akurasi tinggi, dan generalisasi yang cukup baik terhadap data baru.

CONFUSION MATRIX



CLASSIFICATION REPORT

```
============================================================
 METRICS PER CLASS (TEST SET)
============================================================
Class           Precision    Recall       F1-Score
------------------------------------------------------------
plane           0.849        0.903        0.875
car             0.947        0.947        0.947
bird            0.841        0.812        0.826
cat             0.762        0.732        0.747
deer            0.846        0.878        0.862
dog             0.810        0.796        0.803
frog            0.892        0.928        0.910
horse           0.925        0.894        0.909
ship            0.928        0.926        0.927
truck           0.923        0.909        0.916
============================================================

 OVERALL METRICS (TEST SET)
============================================================
Type            Precision    Recall       F1-Score
------------------------------------------------------------
Macro-Average   0.872        0.873        0.872
Weighted-Average 0.872        0.873        0.872
============================================================
```

RESNET

ARSITEKTUR

```python
print("\nMembuat model ResNet18 yang dimodifikasi untuk CIFAR-10...")  # Informasi proses pembuatan model

if optimized_hyperparameters["train_from_scratch"]:  # Jika flag pelatihan dari awal diaktifkan
    resnet = models.resnet18(weights=None, num_classes=num_classes)  # Membuat ResNet18 tanpa bobot pretrained
    print("Melatih ResNet18 dari awal (weights=None).")  # Cetak informasi
else:
    resnet = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)  # Load ResNet18 pretrained ImageNet
    print("Menggunakan ResNet18 dengan bobot pretrained ImageNet.")  # Cetak informasi
    num_ftrs = resnet.fc.in_features  # Ambil jumlah fitur dari layer fully connected (fc)
    resnet.fc = nn.Linear(num_ftrs, num_classes)  # Ganti fc layer agar sesuai dengan jumlah kelas CIFAR-10

# Modifikasi awal ResNet agar sesuai dengan ukuran gambar CIFAR-10 (32x32)
resnet.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1, bias=False)  # Ganti layer konvolusi pertama
resnet.maxpool = nn.Identity()  # Hilangkan maxpooling untuk mempertahankan resolusi gambar

# Tambahkan dropout jika dilatih dari awal dan nilai dropout > 0
if optimized_hyperparameters["train_from_scratch"] and optimized_hyperparameters["dropout_rate"] > 0:
    if hasattr(resnet, 'fc') and isinstance(resnet.fc, nn.Linear):  # Pastikan fc adalah linear layer
        num_ftrs = resnet.fc.in_features  # Ambil jumlah fitur input dari fc layer lama
        resnet.fc = nn.Sequential(  # Ganti fc layer dengan Dropout + Linear
            nn.Dropout(optimized_hyperparameters["dropout_rate"]),  # Tambahkan dropout
            nn.Linear(num_ftrs, num_classes)  # Tambahkan linear layer
        )
        print(f"FC layer diganti dengan Dropout ({optimized_hyperparameters['dropout_rate']}) dan Linear.")  # Cetak info

```

ARSITEKTUR

Lapisan	Input Shape	Output Shape	Parameter Utama	Jumlah Parameter	Keterangan
conv1	(3, 32, 32)	(64, 32, 32)	Kernel: 3x3, Filters: 64, Stride: 1, Padding: 1, Bias: False	1,728	Konvolusi awal untuk ekstraksi fitur CIFAR-10
batch_norm1	(64, 32, 32)	(64, 32, 32)	Channels: 64	128	Normalisasi setelah konvolusi pertama
relu	(64, 32, 32)	(64, 32, 32)	-	0	Aktivasi non-linear
maxpool	(64, 32, 32)	(64, 32, 32)	-	0	Tidak ada pooling (diganti nn.Identity)
layer1 (BasicBlock 1)	(64, 32, 32)	(64, 32, 32)	2x Conv2d(64, 64, 3x3, stride=1, padding=1), BatchNorm, ReLU	73,984	Blok residual pertama (2 konvolusi)
layer1 (BasicBlock 2)	(64, 32, 32)	(64, 32, 32)	2x Conv2d(64, 64, 3x3, stride=1, padding=1), BatchNorm, ReLU	73,984	Blok residual kedua
layer2 (BasicBlock 1)	(64, 32, 32)	(128, 16, 16)	1st Conv: (64, 128, 3x3, stride=2, padding=1), 2nd Conv: (128, 128, 3x3), BatchNorm, ReLU, Downsample: Conv2d(64, 128, 1x1, stride=2)	229,888	Blok residual dengan downsampling
layer2 (BasicBlock 2)	(128, 16, 16)	(128, 16, 16)	2x Conv2d(128, 128, 3x3, stride=1, padding=1), BatchNorm, ReLU	295,424	Blok residual
layer3 (BasicBlock 1)	(128, 16, 16)	(256, 8, 8)	1st Conv: (128, 256, 3x3, stride=2, padding=1), 2nd Conv: (256, 256, 3x3), BatchNorm, ReLU, Downsample: Conv2d(128, 256, 1x1, stride=2)	919,04	Blok residual dengan downsampling
layer3 (BasicBlock 2)	(256, 8, 8)	(256, 8, 8)	2x Conv2d(256, 256, 3x3, stride=1, padding=1), BatchNorm, ReLU	1,181,696	Blok residual
layer4 (BasicBlock 1)	(256, 8, 8)	(512, 4, 4)	1st Conv: (256, 512, 3x3, stride=2, padding=1), 2nd Conv: (512, 512, 3x3), BatchNorm, ReLU, Downsample: Conv2d(256, 512, 1x1, stride=2)	3,672,064	Blok residual dengan downsampling
layer4 (BasicBlock 2)	(512, 4, 4)	(512, 4, 4)	2x Conv2d(512, 512, 3x3, stride=1, padding=1), BatchNorm, ReLU	4,720,640	Blok residual
avgpool	(512, 4, 4)	(512, 1, 1)	Output size: (1, 1)	0	Pooling untuk merata-rata fitur secara spasial
flatten	(512, 1, 1)	512	-	0	Meratakan output untuk lapisan FC
fc	512	10	Input: 512, Output: 10	5,13	Output logits untuk 10 kelas CIFAR-10

HYPERPARAMETER

Parameter yang di-tuning:

- Learning Rate (lr): [0.1]
- Batch Size (batch_size): [128]

Optimizer: SGD

SGD dengan momentum (0.9) dan weight decay (5e-4) untuk regularisasi, memungkinkan konvergensi yang lebih stabil.

Scheduler: CosineAnnealingLR

Mengurangi learning rate secara bertahap menyerupai fungsi kosinus selama 75 epoch, untuk meningkatkan performa pelatihan.

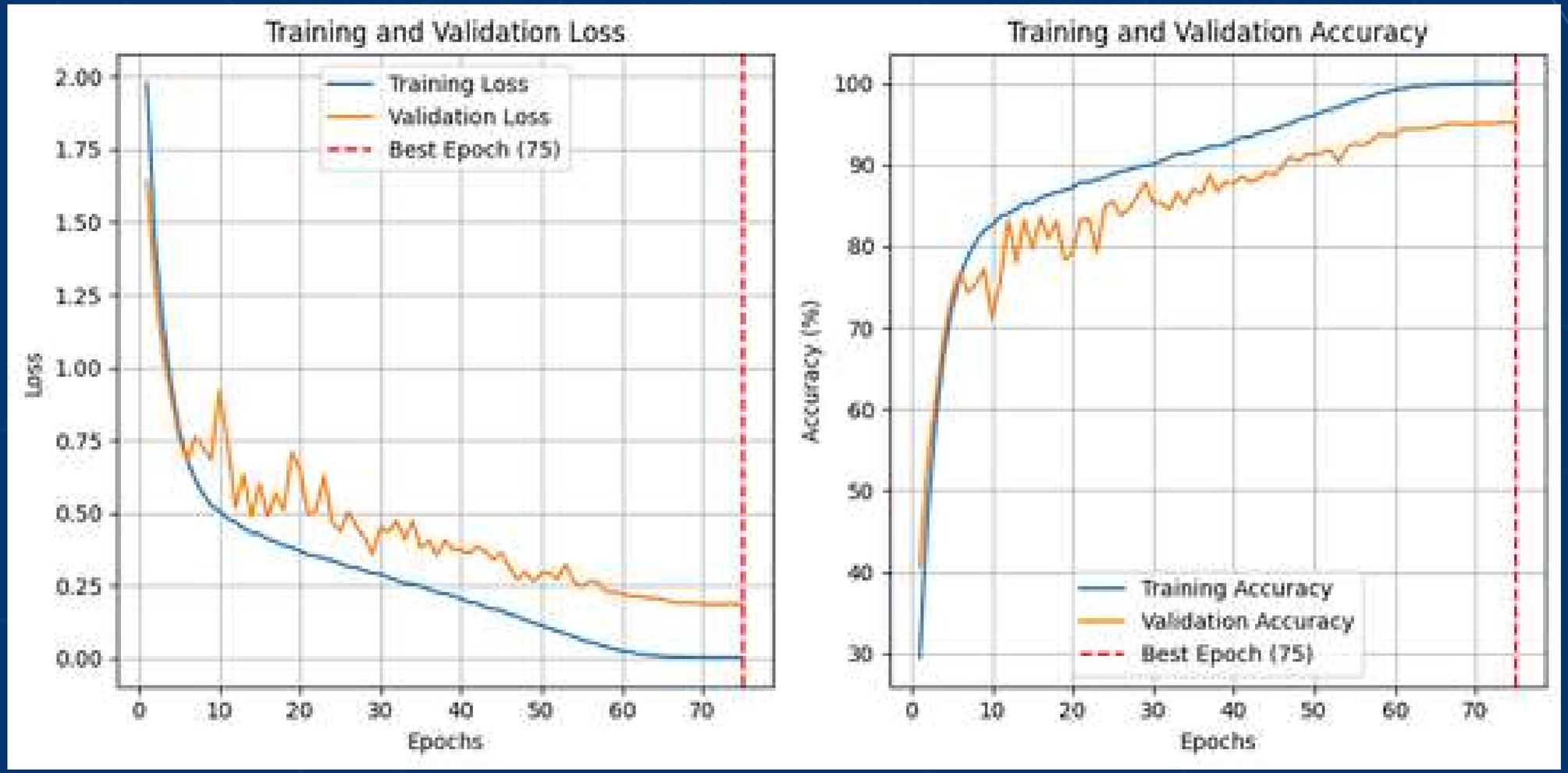
Epochs: 75, Early Stopping: Patience=15

75 epoch dengan early stopping (patience=15) untuk menghentikan pelatihan jika akurasi validasi stagnan.

Hasil:

Best lr: 0.1, Best batch_size: 128 (Val Acc: 94.64%)

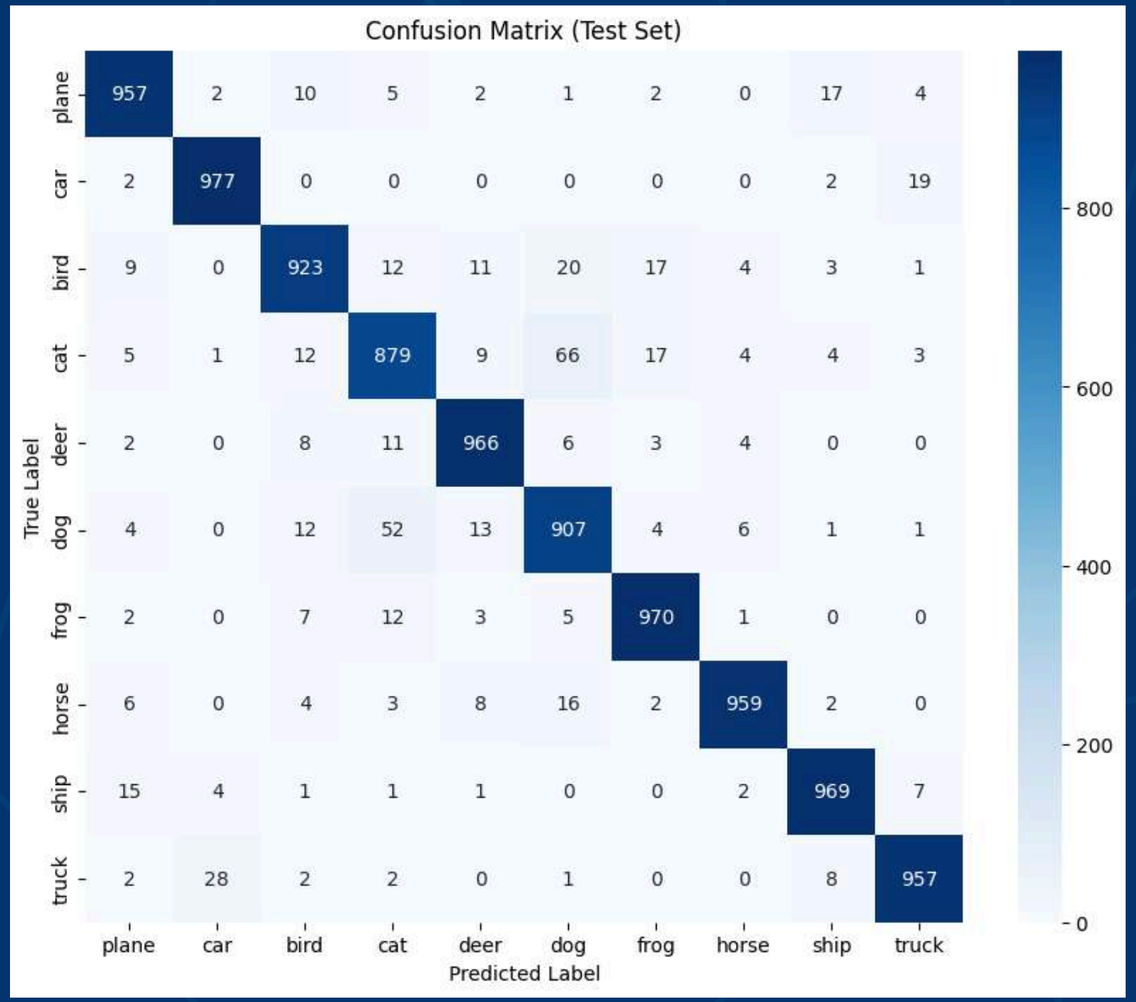
PLOTTING LOSS & AKURASI



PLOTTING LOSS & AKURASI

1. Plot "Training and Validation Loss"
   - Garis biru (Training Loss): loss di data pelatihan terus menurun secara konsisten dari awal hingga akhir epoch — menandakan model belajar dengan baik dari data training.
   - Garis oranye (Validation Loss): Turun drastis di awal, kemudian mengalami fluktuasi kecil namun tetap stabil turun hingga akhir.
   - Garis merah putus-putus menandakan "Best Epoch (75)", yaitu epoch di mana model mencapai performa terbaik pada data validasi.
   - Model tidak overfitting karena training loss dan validation loss sama-sama menurun, dan jaraknya tetap kecil hingga akhir pelatihan. Ini menunjukkan generalisasi yang baik.
2. Plot "Training and Validation Accuracy"
   - Training Accuracy (biru) terus meningkat dan mendekati 100%, menunjukkan model sangat memahami data latih.
   - Validation Accuracy (oranye) juga meningkat, mencapai sekitar 94% di akhir, tanpa penurunan drastis.
   - Model tidak mengalami overfitting yang signifikan karena akurasi validasi juga meningkat dan tetap mendekati akurasi training.

CONFUSION MATRIX



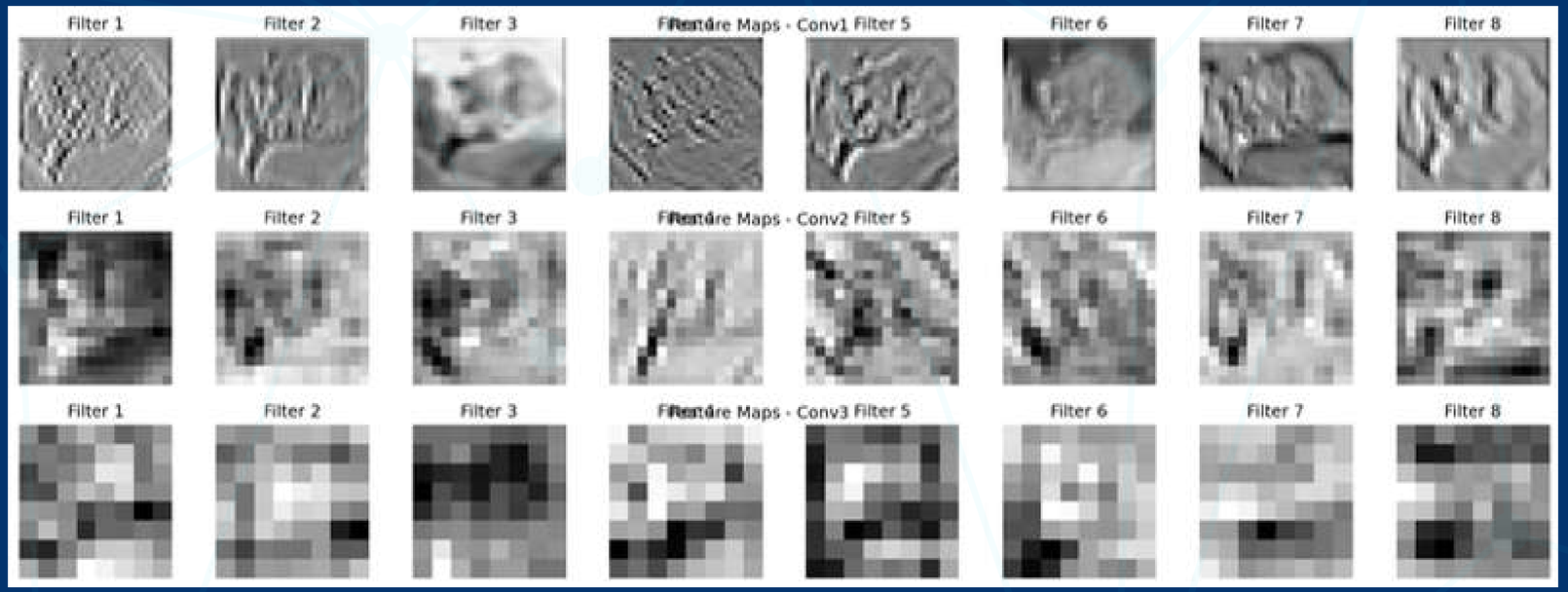
CLASSIFICATION REPORT

```
--- Classification Report (Test Set) ---
              precision    recall  f1-score   support

       plane       0.95      0.96      0.96      1000
         car       0.97      0.98      0.97      1000
        bird       0.94      0.92      0.93      1000
         cat       0.90      0.88      0.89      1000
        deer       0.95      0.97      0.96      1000
         dog       0.89      0.91      0.90      1000
        frog       0.96      0.97      0.96      1000
       horse       0.98      0.96      0.97      1000
        ship       0.96      0.97      0.97      1000
       truck       0.96      0.96      0.96      1000

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000


--- Accuracy per Class ---
 plane: 95.70%
   car: 97.70%
  bird: 92.30%
   cat: 87.90%
  deer: 96.60%
   dog: 90.70%
  frog: 97.00%
 horse: 95.90%
  ship: 96.90%
 truck: 95.70%
```

FEATURE MAPS

CNN

Ground Truth: cat
Predicted: cat

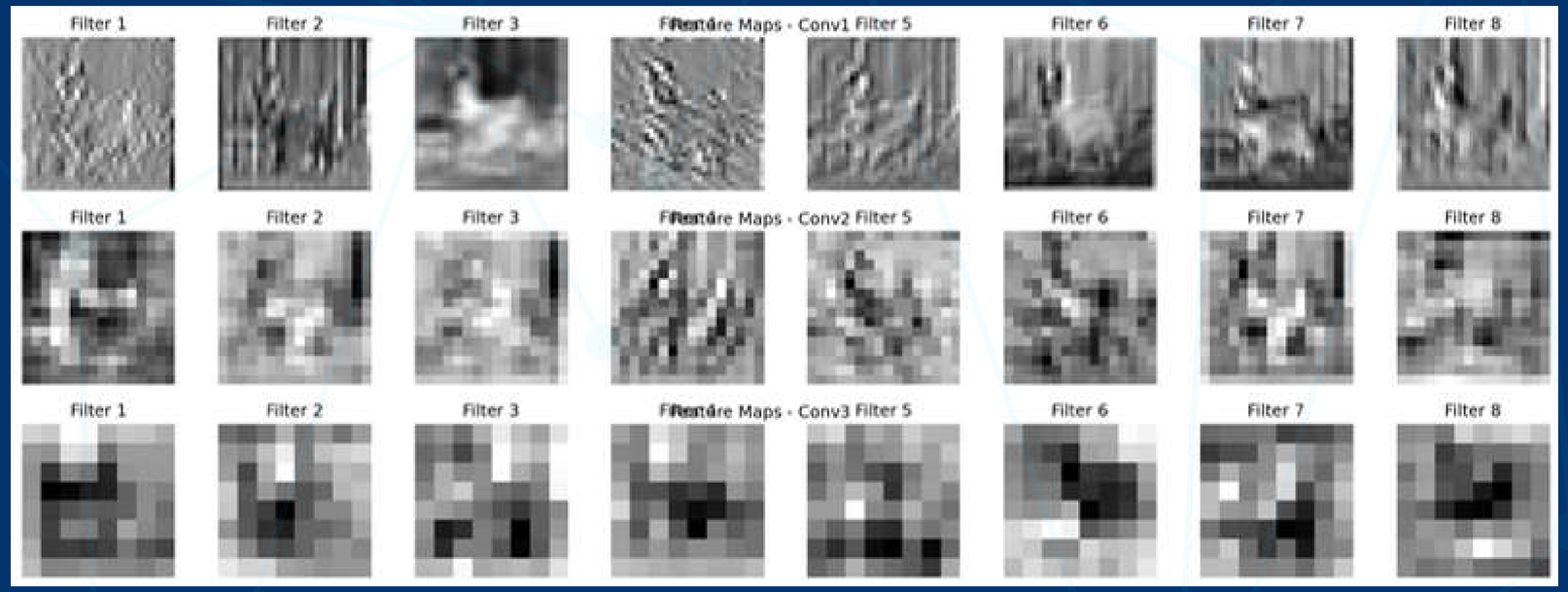


Kesimpulan:

- Menunjukkan bahwa CNN berhasil membangun hierarki fitur, dari sederhana hingga kompleks.
- Aktivasi di tiap layer menunjukkan respons kuat pada bagian-bagian penting gambar, seperti bentuk wajah kucing atau pola bulu.
- Kombinasi aktivasi yang baik ini membantu model mengenali objek secara akurat, terbukti dari prediksi yang benar.

CNN

Ground Truth: dog
Predicted: deer

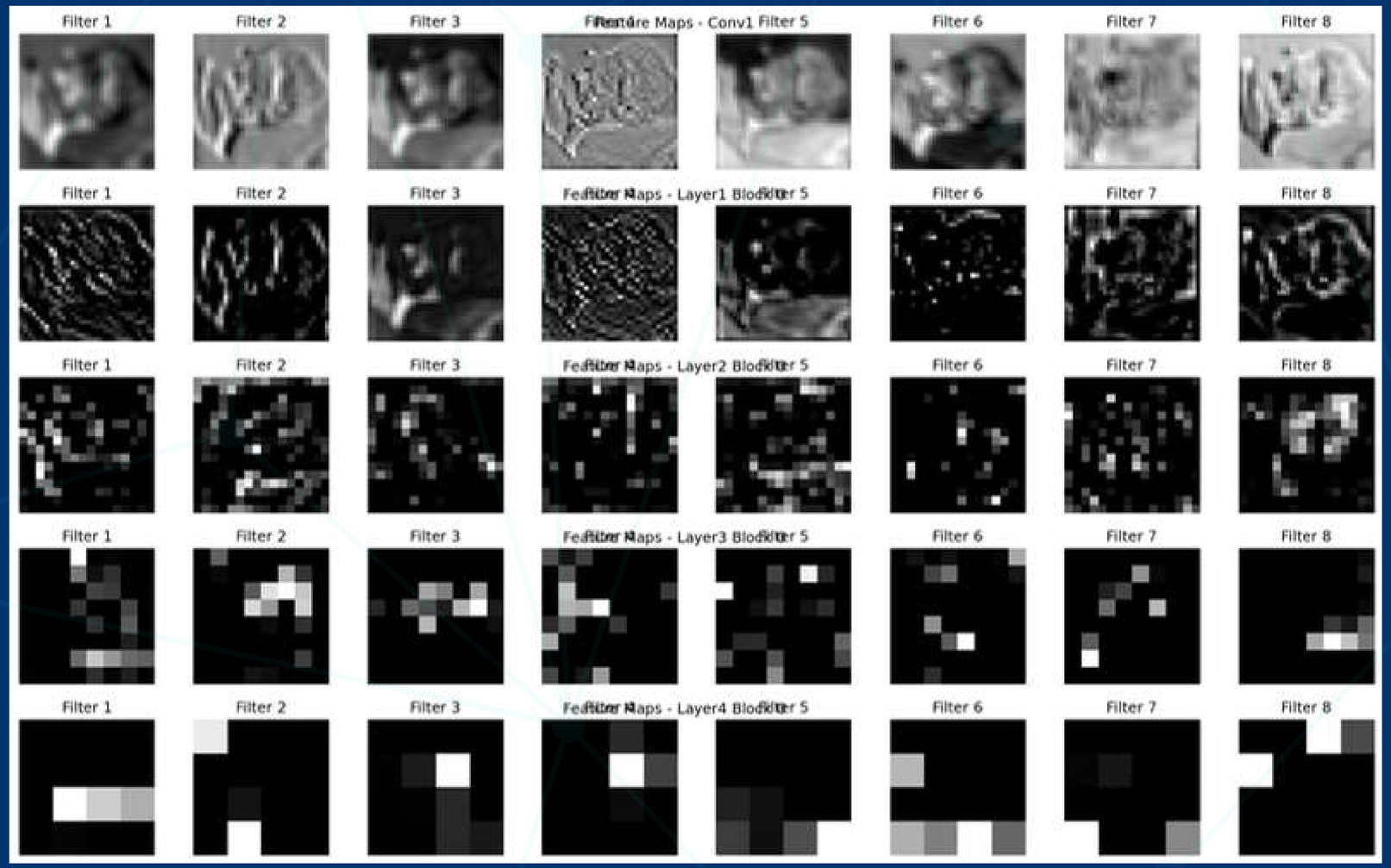


Kesimpulan:

- Menunjukkan bahwa CNN gagal menangkap hierarki fitur
- Aktivasi awal menunjukkan respons terhadap pola tekstur.
- Namun, di layer akhir: Aktivasi kurang fokus pada fitur khas anjing (wajah, telinga). Pola lebih menyerupai struktur umum yang bisa dimiliki deer.
- Kemungkinan penyebab kesalahan: Kemiripan visual dengan deer (background, postur), kurangnya representasi gambar anjing dalam data pelatihan.

RESNET

Ground Truth: cat
Predicted: cat

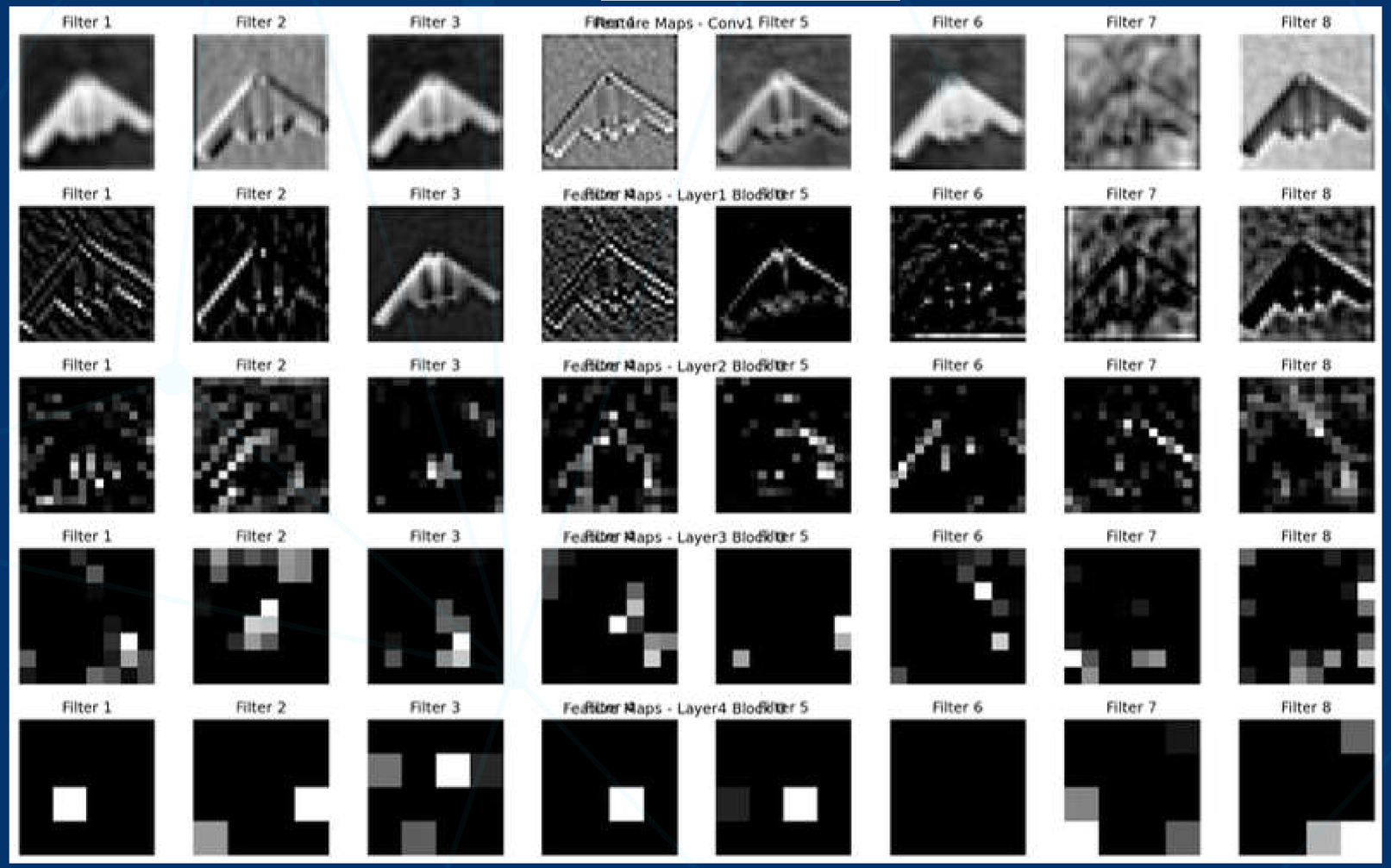


Kesimpulan:

- Model benar-benar memfokuskan diri pada karakteristik khas seekor kucing.
- Aktivasi pada lapisan akhir sangat spesifik dan menunjukkan kepercayaan model yang tinggi terhadap fitur-fitur penting.
- Hal ini merupakan indikator positif bahwa Resnet telah berhasil mempelajari representasi internal yang relevan untuk tugas klasifikasi.

RESNET

Ground Truth: plane
Predicted: dog



Kemungkinan Penyebab Mispredikси:

- Model kehilangan representasi khas pesawat di layer yang lebih dalam.
- Terdapat kemiripan struktural antara gambar pesawat ini dan fitur umum anjing (bisa jadi background atau pose).
- Kemungkinan overfitting pada data anjing atau underrepresentation kelas pesawat di data pelatihan.

PERBANDINGAN

AKURASI

Model	CNN	Resnet
Training Accuracy	89.58%	99.97%
Validation Accuracy	86.84%	95.26%
Test Accuracy	87.25%	96.64%

KINERJA MODEL

Metrik	CNN	Resnet
Accuracy	87.25%	94.64%
Precision (macro)	87.20%	95%
Recall (macro)	87.30%	95%
F1 Score (macro)	87.20%	95%

Label	CNN	Resnet
Plane	90.30%	95.70%
Car	94.70%	97.70%
Bird	81.20%	92.30%
Cat	73.20%	87.90%
Deer	87.80%	96.60%
Dog	79.60%	90.70%
Frog	92.80%	97.00%
Horse	89.40%	95.90%
Ship	92.60%	96.90%
Truck	90.90%	95.70%

AKURASI PER KELAS

LOG TRAINING MODEL

Aspek	CNN	Resnet
Train Accuracy	89.59%	99.97%
Val Accuracy	86.84%	95.26%
Train Loss	Menurun stabil hingga ~0.3	Menurun drastis dan stabil hingga < 0.05
Val Loss	Menurun, sempat stagnan, lalu stabil turun	Menurun drastis, fluktuasi kecil, lalu stabil
Overfitting	Tidak overfitting	Tidak tampak signifikan
Stabilitas	Stabil dengan sedikit fluktuasi pada val	Sangat stabil hingga akhir
Kesimpulan	Model cukup baik dan generalisasi oke	Model sangat kuat, akurasi dan generalisasi tinggi

KESIMPULAN

KESIMPULAN PERBANDINGAN

Aspek	CNN	Resnet
Akurasi test	87%	94.64%
Kedalaman jaringan	Dangkal (8 conv layer)	Dalam (18 layer dengan residual block)
Kemampuan generalisasi	Cukup baik	Sangat baik
Overfitting	Tidak overfitting	Tidak tampak signifikan
Analisis feature map	Menangkap fitur dasar	Menangkap fitur kompleks dan fokus tinggi
Arsitektur	Custom ringan	Resnet dengan skip connection

Model CNN dan ResNet sama-sama mampu mengklasifikasikan gambar CIFAR-10 secara efektif

- CNN: Akurasi test mencapai 87%
- ResNet: Akurasi validasi mencapai 94.64%

Tidak ditemukan overfitting signifikan pada kedua model

- Grafik loss dan akurasi menunjukkan tren yang stabil.
- Jarak antara training dan validation kecil.

ResNet secara konsisten mengungguli CNN dari segi akurasi, generalisasi, dan kedalaman pembelajaran fitur

- Ditunjukkan dari training curve dan hasil evaluasi pada feature maps.

Optimasi hyperparameter (lr, batch size, scheduler, optimizer) berperan besar dalam peningkatan performa.

- CNN: Adam, lr=0.001, batch=32, ReduceLROnPlateau
- ResNet: SGD, lr=0.1, batch=128, CosineAnnealingLR

Analisis feature maps menunjukkan bahwa kedua model mampu membangun representasi fitur hierarkis, namun ResNet lebih fokus dan kuat dalam membedakan karakteristik kelas.

THANK YOU