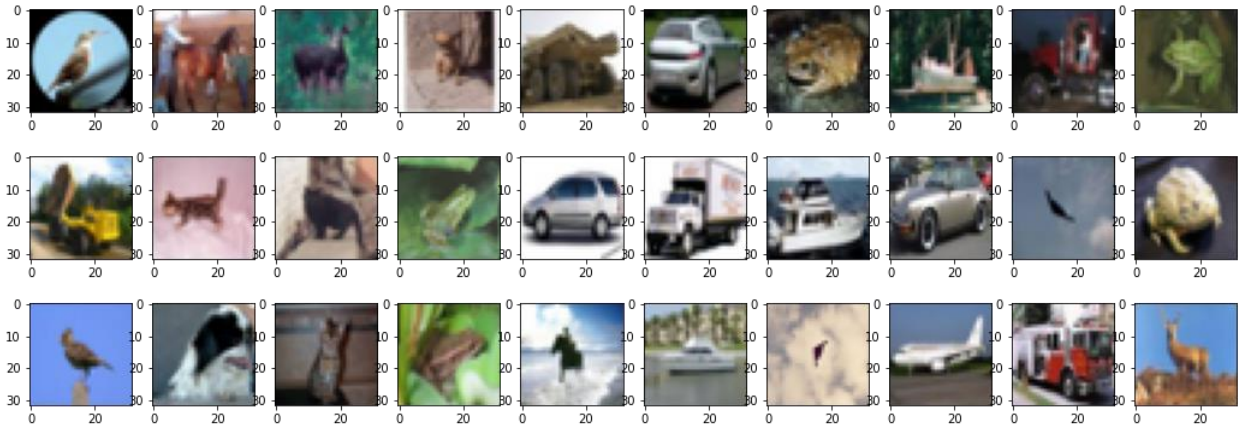
Laporan Tugas Eksplorasi Hyperparameter CNN dan Neural Network

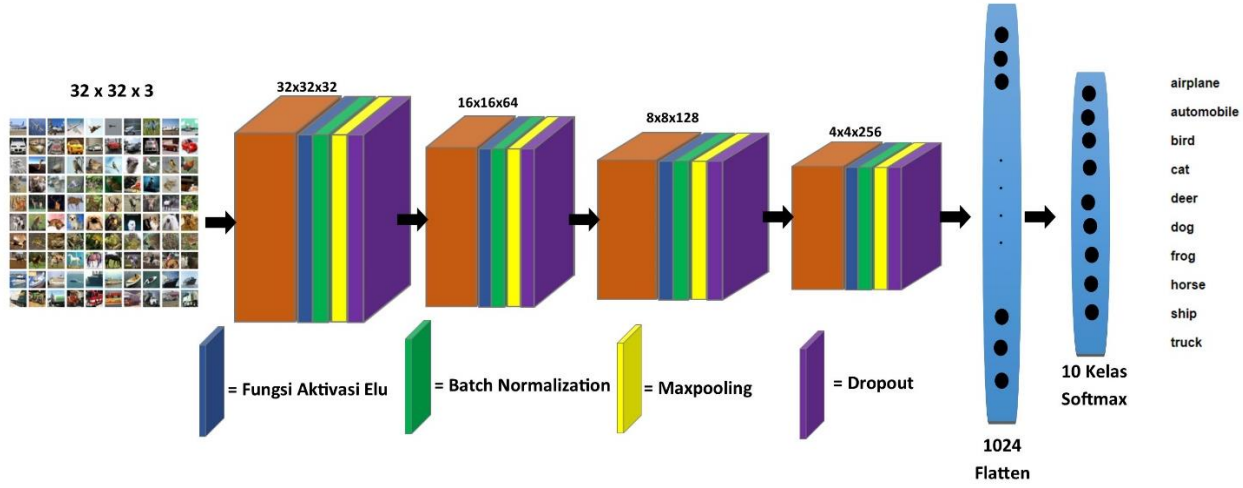
A. Persoalan Klasifikasi

Dataset yang digunakan dalam proses Ekspolarasi Hyperparameter CNN untuk tugas klasifikasi ini adalah CIFAR10 dengan ukuran citra 32x32 dan jumlah kelas sebanyak 10 kelas (*airplane, automobile, bird, cat, deer, dog, frog, horse, ship,* dan *truck*), ditunjukkan pada Gambar 1. Dataset CIFAR10 dibagi menjadi tiga set yaitu data pelatihan (32000), data validasi (8000), dan data tes (10000).



Gambar 1. Contoh Input Citra pada CIFAR10

Proses mendapatkan model CNN yang optimal dimulai dengan menetapkan arsitektur dan nilai *hyperparameter* awal yang akan digunakan. Aristektur awal yang digunakan memanfaatkan 4 layer CNN, Satu Flatten Layer dan Output layer. Setiap layer CNN yang digunakan terdiri dari beberapa operasi sebagai berikut: fungsi aktivasi Elu, *Batch Normalization*, *Maxpooling,* dan *Dropout*. Ilustrasi aristektur awal yang digunakan ditunjukkan pada Gambar 2. Jumlah parameter dari arsitektur awal ini adalah 12170 paramater.



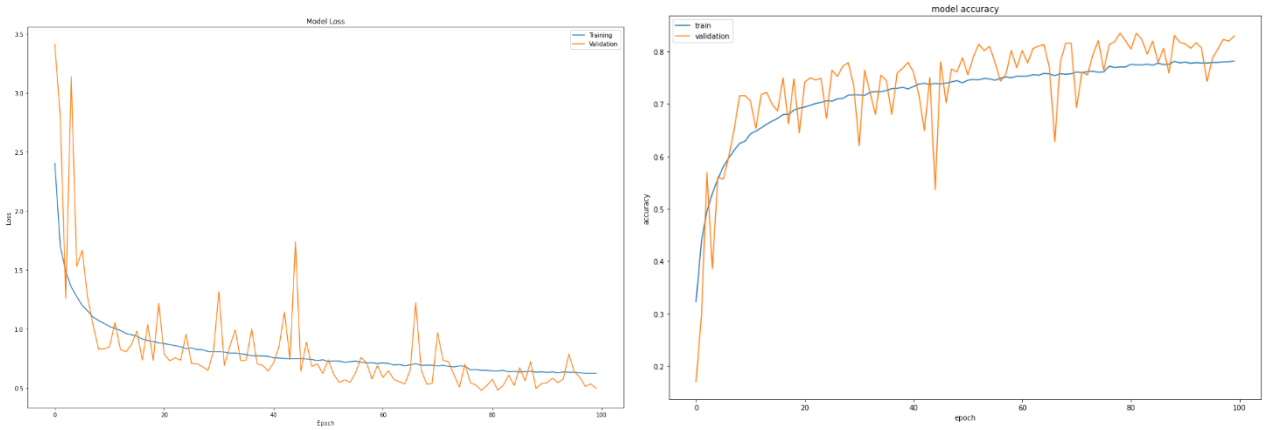
Gambar 2. Arsitektur awal CNN untuk Klasifikasi dataset CIFAR10

Hyperparameter awal yang digunakan untuk proses training ditunjukkan pada tabel 1 dibawah ini:

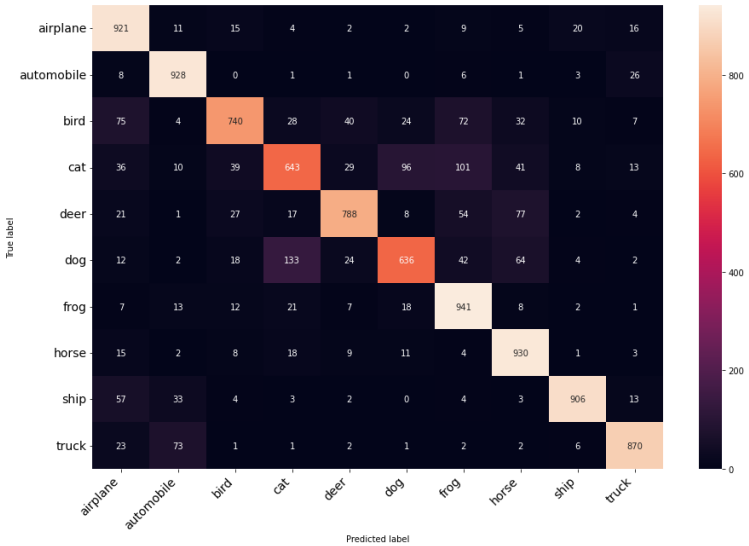
Tabel 1. Nilai Hyperparameter awal

No	*Hyperparameter*	Nilai
1	Ukuran Filter Konvolusi	3x3
2	Fungsi Aktivasi	Elu
3	Jumlah Filter	32 pada Layer 1, 64 pada Layer 2, 128 pada Layer 3, 256 pada Layer 4
4	Batch Size	128
5	Optimizer	Rmsprop
6	Loss Function	Categorical Crossentropy
6	Epoch	100
7	Learning Rate	0.001
8	LR Scheduler	LR scheduler di set awalnya sebesar 0.001 setelah epoch > 75 LR-nya menjadi 0.0005
9	Hidden unit	1024 (Hasil Flatten dari layer 4)
10	Jumlah Layer CNN	4

Pada proses training menunjukkan nilai akurasi memiliki kecenderungan terus meningkat seiring pertambahan epoch dan sebaliknya nilai lossnya terus berkurang, seperti ditunjukkan pada Gambar 3. Pada percobaan awal ini didapatkan nilai akurasi untuk tes set sebesar 83.03 % dengan loss sebesar 0.499.

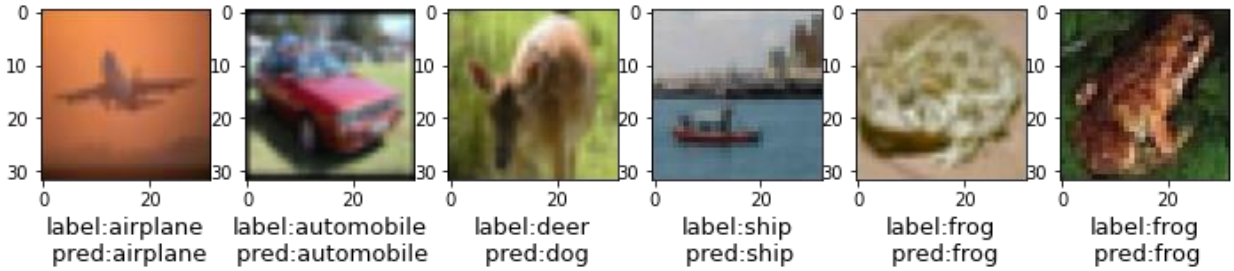


Gambar 3. Grafik loss (kiri) dan akurasi (kanan) untuk percobaan awal



Gambar 4. *Confussion matrix* percobaan awal

Selain itu dari percobaan awal didapat grafik confusion matrix untuk mengetahui keakuratan model yang digunakan dalam mengklasifikasikan tiap kelasnya. Seperti yang ditunjukkan pada Gambar 4. Sebagai contoh pada model awal ini didapatkan bahwa kelas "dog" dan "cat" merupakan kelas yang memiliki akurasi yang rendah. Selanjutnya dibawah ini adalah contoh tampilan output gambar yang benar dan salah untuk percobaan awal ini.



Gambar 5. Ouptut hasil percobaan awal.

A.1 Eksplorasi Jumlah Convolutional layer yang optimal.

Pada percobaan pertama ini untuk mengetahui berapa banyak jumlah layer yang optimal untuk klasifikasi CIFAR10, dilakukan modifikasi arsitektur awal dengan menggunakan jumlah layer CNN yang berbeda. Jumlah layer yang digunakan dalam eksplorasi ini adalah 4 (jumlah Layer pada arsitektur awal), 8, 16, dan 32. Untuk Hyperparameter yang lain dibuat tetap seperti pada Tabel 1. Dari hasil simulasi pada data uji didapatkan hasil seperti yang ditunjukkan pada Tabel 2.

Tabel 2. Hasil Eksplorasi Jumlah Layer CNN yang berbeda

Jumlah Layer	Jumlah Parameter	Loss	Akurasi (%)
4 layer CNN	400.586	0.499	83.03
8 layer CNN	1.186.346	0.463	85.85
16 layer CNN	**2.757.866**	**0.4012**	**86.77**
32 layer CNN	5.900.906	0.555	81.65

Dari hasil simulasi didapatkan bahwa arsitektur dengan 16 layer CNN menghasilkan hasil yang optimal (akurasi 86.77% dan loss 0.4012) dibandingkan dengan jumlah layer yang lain. Pada percobaan ini memperlihatkan bahwa penambahan layer CNN tidak selalu berbanding lurus dengan peningkatan akurasi dari model yang diuji. Khusus untuk jumlah layer 32, dari percobaan dilakukan nilai akurasinya semakin meningkat setiap epoch-nya. Namun, proses konvergensinya sangat lambat karena kompleksitas dan jumlah parameternya yang semakin banyak. Sehingga ada kecenderungan model ini overfitting yang mengakibatkan penurunan akurasi pada data uji.

A.2 Eksplorasi Ukuran Filter untuk setiap Convolutional layer yang optimal.

Selanjutnya untuk mengetahui ukuran filter yang optimal untuk setiap Convolution layer, dilakukan percobaan dengan menggunakan ukuran filter yang berbeda. Melanjutkan percobaan A.1, ukuran filter yang berbeda akan digunakan pada aristektur yang sebelumnya yaitu 16 layer CNN. Ukuran filter yang digunakan yaitu 3x3 (ukuran filter pada arsitektur awal), 5x5, 7x7, dan 9x9. Alasan memilih ukuran filter yang ganjil karena semua piksel lapisan sebelumnya akan simetris di sekitar piksel keluaran. Tanpa simetri ini, harus memperhitungkan distorsi di seluruh lapisan yang terjadi saat menggunakan kernel berukuran

genap. Nilai hyperparameter yang lain dibuat tetap seperti Tabel 1, kecuali jumlah layer yang berubah dari 4 menjadi 16 disesuai dengan hasil simulasi A.1. Dari hasil simulasi pada data uji didapatkan hasil seperti yang ditunjukkan pada Tabel 3.

Tabel 3. Hasil Eksplorasi Ukuran Filter yang berbeda pada Convolutional Layer

Ukuran Filter	Jumlah Parameter	Loss	Akurasi (%)
3x3	**2.757.866**	**0.4012**	**86.77**
5x5	7.625.450	0.440	86.03
7x7	14.926.826	0.520	83.22
9x9	24.661.994	0.522	83.03

Dari hasil simulasi didapatkan bahwa filter dengan ukuran 3x3 menghasilkan nilai akurasi yang optimal (86.77%) dibandingkan dengan ukuran filter lainnya. Ditambah lagi, model ini mendapatkan akurasi tersebut dengan jumlah parameter yang paling sedikit dibandingkan dengan ukuran filter lainnya. Selain itu, percobaan ini menunjukkan bahwa penambahan jumlah filter mengakibatkan penambahan jumlah parameter secara kuadratis yang mengakibatkan proses komputasi yang semakin kompleks. Ditambah lagi nilai akurasi juga ikut menurun seiring bertambahnya ukuran filter, hal ini kemungkinan bisa disebabkan karena proses *padding* sehingga banyak "unrelated features" (nilai nol) yang ikut diproses pada konvolusinya.

A.3 Eksplorasi Jumlah Filter yang optimal untuk setiap Convolutional layer.

Ekplorasi yang selanjutnya adalah menentukan jumlah filter yang optimal untuk setiap convolution layer. Ekplorasi ini melanjutkan simulasi A.1 dan A.2 dimana kita mendapatkan 16 layer CNN dengan ukuran filter 3x3 untuk setiap convolution layernya. Seperti yang diketahui untuk ukuran filter ini berdasarkan hyperparameter awal pada Tabel 1, diatur untuk setiap layernya berbeda yaitu 32 pada Layer 1, 64 pada Layer 2, 128 pada Layer 3, 256 pada Layer 4. Karena yang digunakan pada percobaan ini disesuaikan dengan hasil simulasi A.1 yaitu 16 layer CNN maka terdapat perubahan jumlah filter sebagai berikut: 32 pada Layer 1-4, 64 pada Layer 5-8, 128 pada Layer 9-12, 256 pada Layer 13-16 Untuk menentukan jumlah filter yang optimal ini, digunakan jumlah filter yang sama untuk setiap layernya yaitu 8, 16, 32, dan 64. Contohnya, apabila jumlah filternya adalah 8 maka dari layer 1 sampai 16 akan diatur sebanyak 8. Dari hasil simulasi pada data uji didapatkan hasil seperti yang ditunjukkan pada Tabel 4.

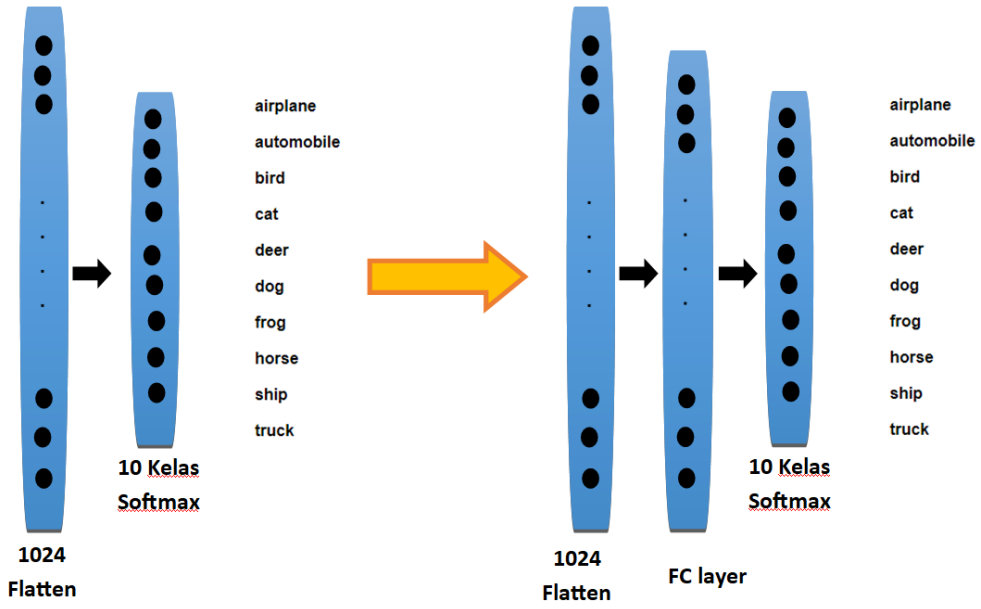
Tabel 4. Hasil Eksplorasi Jumlah Filter yang berbeda pada Convolutional Layer

Jumlah Filter	Jumlah Parameter	Loss	Akurasi (%)
32 pada Layer 1-4, 64 pada Layer 5-8, 128 pada Layer 9-12, 256 pada Layer 13-16	**2.757.866**	**0.4012**	**86.77**
8 pada layer 1-16	9.826	1.351	48.640
16 pada layer 1-16	36.992	0.982	66.15
32 pada layer 1-16	142.954	0.557	81.54
64 pada layer 1-16	562.378	0.413	86.67

Dari Hasil Simulasi didapatkan bahwa ukuran filter mempengaruhi akurasi yang didapatkan. Terdapat kecenderungan bahwa semakin banyak jumlah filter yang digunakan akurasi yang didapatkan semakin

meningkat. Selain itu, penambahan jumlah filter mengakibatkan penambahan jumlah parameter tetapi tidak sebanyak dengan penambahan ukuran filter sehingga tidak memebani proses komputasi. Hasil ini juga menunjukkan dengan menggunakan 64 filter untuk setiap layer konvolusi dari 1 sampai 16 mendapatkan akurasi 86.67% dimana berbeda sekitar 0.03% dengan model yang memiliki jumlah filter yang berbedea di tiap layer nya. Namun, apabila dilihat dari jumlah parameternya, model dengan 64 filter disetiap layernya memilki jumlah parameter hampir 5x lebih sedikit dibanding dengan model yang jumlah filternya berbeda di setiap layernya.

A.4 Eksplorasi Jumlah hidden unit yang optimal untuk Fully Connected layer.



Gambar 6. Penambahan satu Fully Connected Layer diantara Flatten dan Output layer.

Pada aristektur awal yang digunakan, fully connected layernya didapatkan dari hasil flatten ouput CNN pada layer terakhir sehingga jumlah hidden unit sangat bergantung pada ukuran output layer CNN terakhir. Oleh karena itu, untuk mengetahui berapa jumlah hidden unit yang optimal ditambahkan 1 layer FC diantara Flatten dan output layer, seperti ditunjukkan pada Gambar 6.

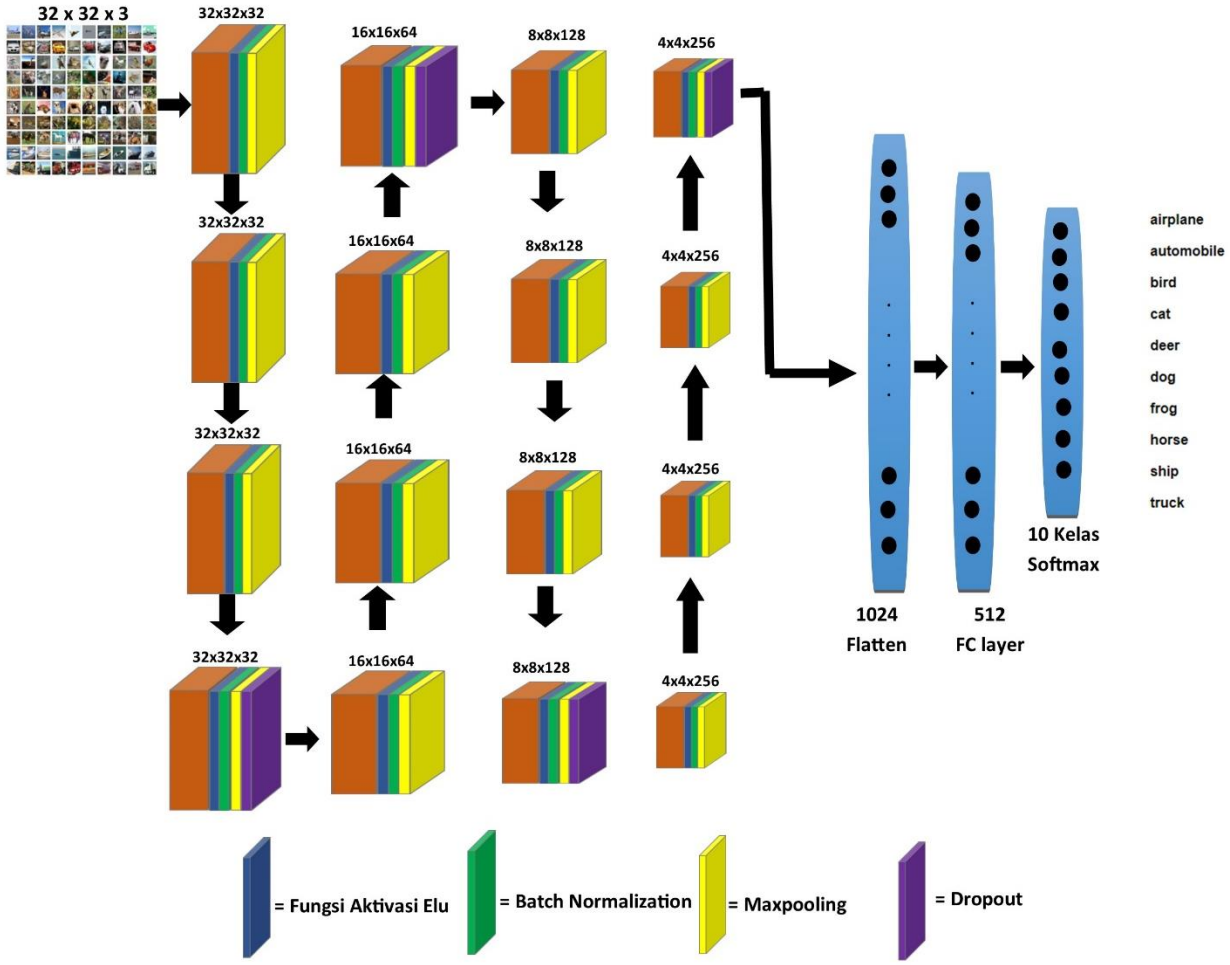
Tabel 5. Hasil Eksplorasi Jumlah Filter yang berbeda pada Convolutional Layer

Jumlah Hidden Units	Jumlah Parameter	Loss	Akurasi (%)
64	2.813.866	0.3867	87.37
128	2.880.106	0.3765	87.52
256	3.012.586	0.383	87.50
512	**3.277.546**	**0.3895**	**87.87**

Pada simulasi ini digunakan hasil yang telah didapatkan dari percobaan A.1 sampai A.3 dimana sejauh ini jumlah layer yang digunakanyaitu 16, ukuran Filter 3x3, dan jumlah filter yang digunakan 32 pada Layer 1-4, 64 pada Layer 5-8, 128 pada Layer 9-12, 256 pada Layer 13-16. Selain parameter tersebut, hyperparameter yang digunakan masih sama dengan pada Tabel 1. Untuk jumlah hidden units yang digunakan adalah sebagai berikut 64, 128, 256, dan 512. Adapun hasil evaluasi untuk jumlah hidden unit

yang optimal ditunjukkan pada Tabel 5. Dari hasil simulasi didapatkan dengan penambahan FC layer meningkatkan akurasi dari model yang digunakan. Terdapat peningkatan kurang lebih hampir 1 % untuk semua hidden units yang dicoba. Jumlah hidden units sebesar 512 mendapatkan hasil yang optimal dibandingkan dengan jumlah hidden units lainnya.

Dari percobaan A.1 sampai A.4 terdapat perubahan aristektur yang dimana model awal menggunakan 4-layer CNN menjadi 16 layer CNN. Selain itu, terdapat penambahan FC layer dengan jumlah hidden units 512. Dimana akurasi yang dihasilkan sebesar 87.87% meningkat sekitar 4.8% dibandingkan dengan arsitektur awalnya. Gambar 7 menunjukkan perubahan aristektur sesuai dengan hasil ekplorasi A.1 sampai A.4.



Gambar 7. Arsitektur akhir CNN untuk Klasifikasi dataset CIFAR10

B.1. Ekplorasi Optimizer yang menghasilkan nilai yang Optimal

Dalam eksplorasi selanjutnya, untuk mengetahui optimizer yang optimal perlu dilakukan percobaan dengan menggunakan optimizer yang telah disediakan oleh keras library. Adapun optimizer yang

digunakan adalah sebagai berikut: RMSprop (hyperparameter awal), SGD, Adam, Adadelta, Adagrad , Adamax, Nadam, dan Ftrl. Learning rate yang digunakan unutk semua optimizer adalah 0.001. Adapun hasil simulasi untuk optimizer yang berbeda ditunjukkan pada Tabel 6.

Tabel 6. Hasil Eksplorasi Optimizer yang disediakan oleh keras library

Optimizer (LR=0.001)	Loss	Akurasi (%)
RMSprop (default)	0.3867	87.87
SGD	2.0428	39.15
Adam	**0.3665**	**88.03**
Adamax	0.4616	85.44
Adagrad	2.3922	27.29
Adadelta	3.8452	13.36
Nadam	0.4126	87.04
Ftrl	2.3026	9.52

Dari hasil simulasi dengan menggunakan optimizer yang berbeda didapatkan bahwa optimizer Adam memperoleh kinerja yang lebih baik dibandingkan dengan optimizer lainnya. Terdapat kenaikan sekitar 0.16% dibandingkan dengan RMSprop yang merupakan optimizer default pada arsitektur awal.

B.2. Ekplorasi learning rate schedule yang menghasilkan kinerja yang lebih baik.

Dalam proses eksplorasi learning rate schedule terdapat 4 pengaturan yang digunakan yaitu sebagai berikut:

```
#LR Scheduler default
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 75:
        lrate = 0.0005
    return lrate
```

```
#LR Scheduler seeting 1
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 35:
        lrate = 0.0006
    elif epoch > 70:
        lrate = 0.0004
    return lrate
```

```
# LR Scheduler seeting 2
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 35:
        lrate = 0.0004
    elif epoch > 70:
        lrate = 0.0002
    return lrate
```

```
#LR Scheduler seeting 3
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 50:
        lrate = 0.0002
    elif epoch > 80:
        lrate = 0.0001
    return lrate
```

Untuk hasil dari penggunanaan LR schedule dapat dilihat pada Tabel 7 dibawah ini:

Tabel 7. Hasil Eksplorasi LR Schedule yang berbeda

LR Schedule Setting	Loss	Akurasi (%)
Default	**0.3665**	**88.03**
Setting 1	0.3948	87.01
Setting 2	0.3804	87.51
Setting 3	0.3750	87.74

Dari hasil simulasi ini bisa dilihat bahwa adanya perubahan LR schedule yang tepat dapat mempengaruhi akurasi dari model yang disimulasikan. Selain itu pemilihan LR untuk setiap syarat epoch memperlihatkan juga pengaruh terhadap kinerja model yang digunakan.

B.3. Ekplorasi losses optimization pada keras (probabilistic) yang memiliki kinerja yang optimal

Fungsi loss pada keras (probabilistic) adalah sebagai berikut: *Binary Crossentropy, Categorical Crossentropy, Sparse Categorical Crossentropy, poisson, dan KL divergence*. Khusus untuk *Categorical Crossentropy* dan Sparse *Categorical Crossentropy* pada simulasi hanya digunakan *Categorical Crossentropy* karena model format labelnya dalam bentuk *one-hot encoded* sedangkan Sparse *Categorical Crossentropy* membutuhkan format label dalam bentuk integer. Adapun hasil simulasi untuk loss function yang berbeda ditunjukkan pada Tabel 8 dibawah ini:

Tabel 8. Hasil Eksplorasi Loss Function yang berbeda

Loss Function	Loss	Akurasi (%)
Categorical Crossentropy (default)	**0.3665**	**88.03**
Binary Crossentropy	0.0646	87.83
Poisson	0.1388	87.61
KL Divergence	**0.3573**	**88.60**

Dari hasil simulasi untuk loss function didapatkan bahwa fungsi KL divergence memeproleh kinerja yang lebih baik dibandingkann dengan loss function lainnya. Terdapat peningkatan sekitar 0.57% dibandingkan dengan loss function *categorical crossentropy* yang merupakan parameter awal.

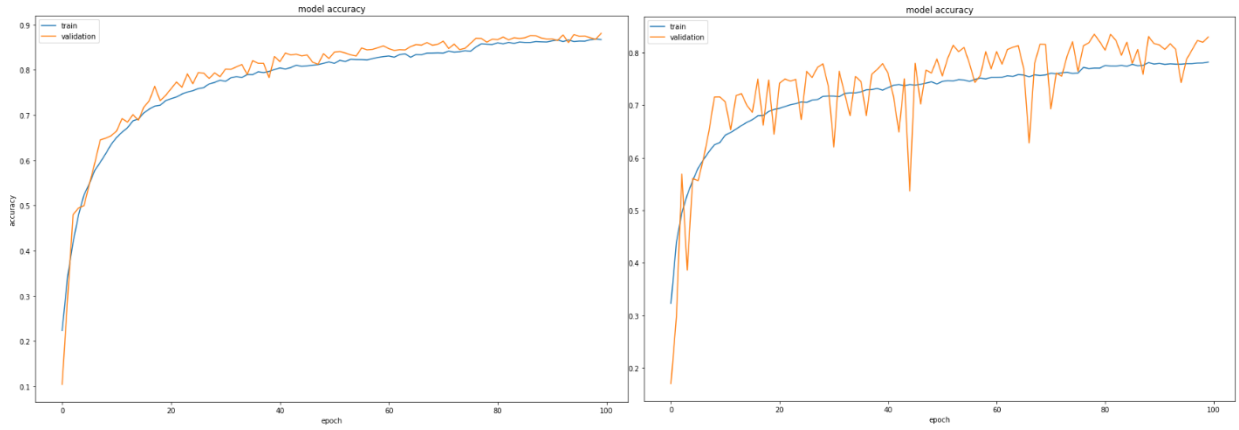
Kesimpulan

Dari hasil eksplorasi beberapa hyperparameter yang disimulasikan terdapat perubahan yang dilakukan dengan meilhat hasil kinerjanya, diantara lain yaitu jumlah layer dari 4 menjadi 16-layer CNN dan penambahan fully connected layer dengan hidden units berjumlah 512. Selain itu, optimizer RMSprop digantikan dengan Adam, serta loss function yang berubah dari Categorical cros entropy menjadi KL divergence. Perubahan lebih lengkapnya dapat dilihat pada Tabel 9, dibawah ini :

Tabel 9. Nilai Hyperparameter akhir

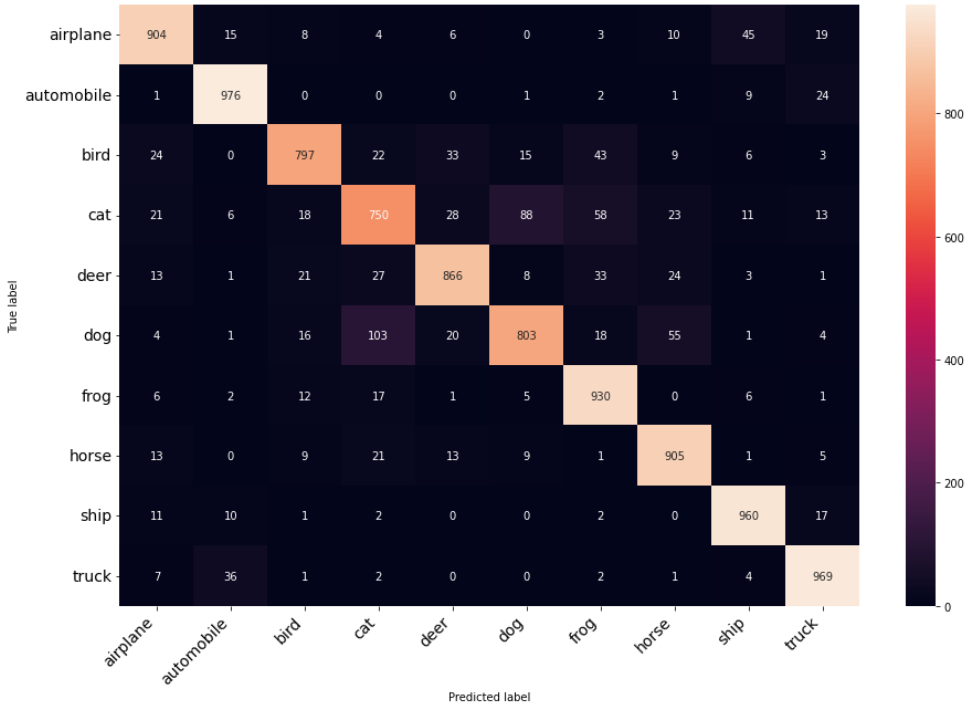
No	***Hyperparameter***	Nilai
1	Ukuran Filter Konvolusi	3x3
2	Fungsi Aktivasi	Elu
3	Jumlah Filter	32 pada Layer 1, 64 pada Layer 2, 128 pada Layer 3, 256 pada Layer 4
4	Batch Size	128
5	**Optimizer**	**Adam**
6	**Loss Function**	**KL Divergence**
6	Epoch	100
7	Learning Rate	0.001
8	LR Scheduler	LR scheduler di set awalnya sebesar 0.001 setelah epoch > 75 LR-nya menjadi 0.0005
9	**Hidden unit**	**512**
10	**Jumlah Layer CNN**	**16**

Pada percobaan terakhir ini didapatkan nilai akurasi pada data uji sebesar 88.60% dimana terdapat peningkatan sekitar 5.57% dari hasil percobaan awal yang dilakukan. Selain itu pada Gambar 8, terlihat perbandingan grafik akurasi untuk percobaan awal dan percobaan akhir dimana percobaan akhir memilki grafik yang lebih halus (smooth) dibanding dengan percobaan awal. Hal ini menunjukkan bahwa proses *hyperparameter tuning* sangat dibutuhkan untk mendapatkan model dengan kinerja yang optimal.

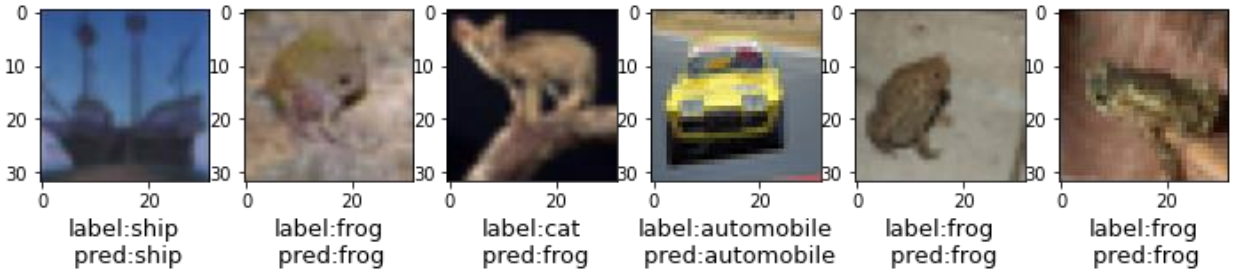


Gambar 8. Grafik akurasi (kanan) untuk percobaan awal dan (kiri) percobaan akhir

Selain itu, dari confusion matrix pada Gambar 9, terjadi peningkatan akurasi dari beberapa kelas khususnya untuk kelas “cat” dan “dog” ada peningkatan yang signifikan. Gambar 10 menunjukkan contoh output yang benar dan salah pada proses klasifikasi.



Gambar 9. Confusion matrix percobaan akhir



Gambar 10. Output percobaan akhir