LAPORAN

EKSPLORASI HYPERPARAMETER CNN DAN NEURAL NETWORK

EI7007 – PEMBELAJARAN MESIN LANJUT

Disusun oleh:

33221036 Mina Ismu Rahayu

(Program Studi Doktor Teknik Elektro dan Informatika)

INSTITUT TEKNOLOGI BANDUNG

Maret 2022

DAFTAR ISI

1. EKSPLORASI HYPERPARAMETER CNN 3
   A. Pemilihan dataset 3
   B. Pra proses data 3
   C. Membangun Arsitektur CNN 3
      Percobaan arsitektur 1 3
      Percobaan Arsitektur 2 5
      Optimizer 7
      Losses 8
   D. Kesimpulan 8
2. EKSPLORASI REGRESI 9
   A. Dataset 9
      Percobaan Arsitektur 1 9
      Percobaan Arsitektur 2 10
      Percobaan Arsitektur 3 10
      Optimizer 11
      Losses 12
   B. Kesimpulan 12

1. EKSPLORASI HYPERPARAMETER CNN

A. Pemilihan dataset

Dataset yang digunakan dalam eksplorasi adalah CIFAR10 yang diambil dari api keras.datasets, cifar10 memiliki 60 ribu gambar dengan ukuran 32 x 32 dengan 10 kategori/ kelas gambar. CIFAR10 membagi gambar menjadi 50 ribu data training dan 10 ribu data testing (Gambar1).

B. Pra proses data

Dalam pra proses data dilakukan pendefenisian nama kelas CIFAR10 yang disimpan kedalam variable array numpy. Selanjutnya dilakukan normalisasi data agar proses training dapat berjalan lebih cepat dengan membagi setiap piksel dalam gambar dengan nilai 255 sehingga range nilai setiap piksel menjadi dalam range 0 sampai 255

Berikut pembagian data didalam CIFAR10

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
(50000, 32, 32, 3) (50000, 1)
(10000, 32, 32, 3) (10000, 1)
```

C. Membangun Arsitektur CNN

Percobaan arsitektur 1

```
cnn_model = tf.keras.Sequential([

    # Layer konvolusi 1 dengan data input 32 x 32

    tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same", activation="relu", input_shape=[32,32,3]),

    #Pendefenisian Max pooling

    tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'),

    # Layer konvolusi 2

    tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same", activation="relu"),

    #Pendefenisian Max pooling 2

    tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'),

    #flatten layer

    tf.keras.layers.Flatten(),

     # mendefenisikan dense layer dengan melakukan aktivasi fugsi RelU

    tf.keras.layers.Dense(128, activation=tf.nn.relu),

    # output dengan melakukan aktifasi fungsi softmax

    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
```

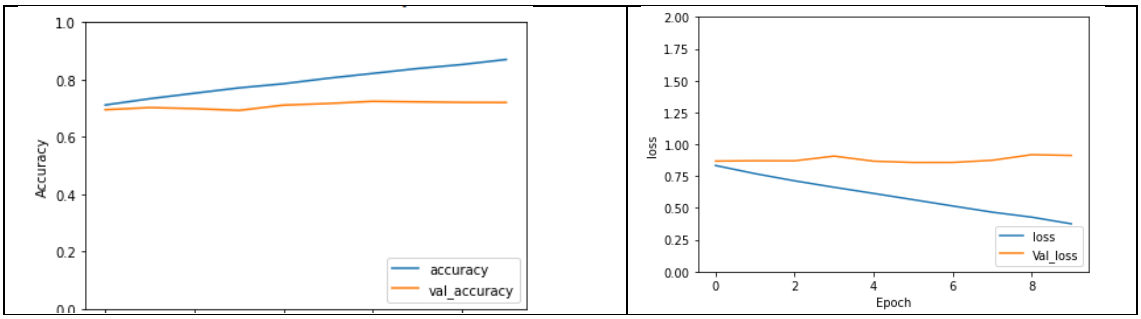
```
])
```

Optimizer yang digunakan adalam Adam optimizer dengan **learning rate 0.001** dan losses yang digunakan **sparse_categorical_crossentropy**

Hasil yang didapatkan dalam proses training **10 epoch** dan batch **128**

loss: 0.3754 - accuracy: 0.8701 - val_loss: 0.9123 - val_accuracy: 0.7200

berikut grafik yang dihasilkan



Percobaan data tesing menggunakan model tersebut terhada random data test **CIFAR10**

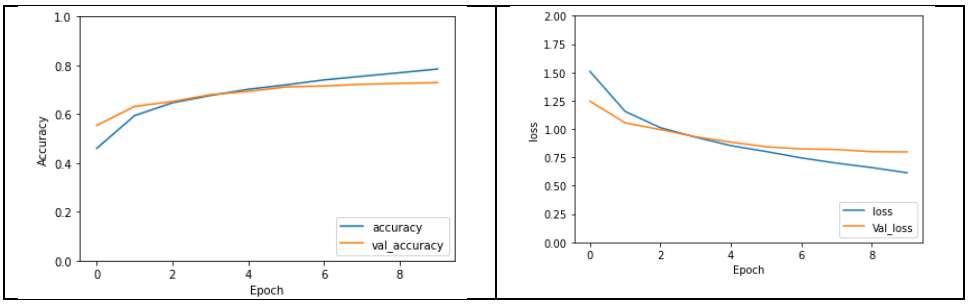
Percobaan selanjutnya untuk mengurang overfit pada val_loss dicoba dengan menambahkan dropout 20% pada dense layer

```
#flatten layer
tf.keras.layers.Flatten(),
#penambahan dropout sebesar 20 persen
 tf.keras.layers.Dropout(0.2,noise_shape=None,seed=None),
 # mendefenisikan dense layer dengan melakukan aktivasi fugsi RelU
 tf.keras.layers.Dense(128, activation=tf.nn.relu),
 # output dengan melakukan aktifasi fungsi softmax
 tf.keras.layers.Dense(10, activation=tf.nn.softmax)
```

Hasil yang didapatkan dalam proses training **10 epoch** dan batch **128**

loss: 0.6139 - accuracy: 0.7849 - val_loss: 0.7983 - val_accuracy: 0.7291

berikut grafik yang dihasilkan



Percobaan data tesing menggunakan model tersebut terhada random data test CIFAR10


```
Jumlah Prediksi benar: 15
Jumlah Prediksi Salah: 5
Rata-rata jawaban benar: 75.0 %
```

Percobaan Arsitektur 2

Percobaan Arsitektur ke 2 mengikuti arsitektur VGG dengan 2 layer konvolusi yang sama pada hidden layer

```
cnn_model = tf.keras.Sequential([
    # Layer konvolusi 1 dengan data input 32 x 32
    tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same", activation="relu", input_shape=[32,32,3]),
    #layer konvolusi 2
    tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same", activation="relu"),
    #Pendefenisian Max pooling 2
    tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'),

    # Layer konvolusi 3
    tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same", activation="relu"),
    # Layer konvolusi 4
    tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same", activation="relu"),
    #Pendefenisian Max pooling 2
    tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'),

    #flatten layer
    tf.keras.layers.Flatten(),
     # mendefenisikan dense layer dengan melakukan aktivasi fugsi RelU
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    # output dengan melakukan aktifasi fungsi softmax
```

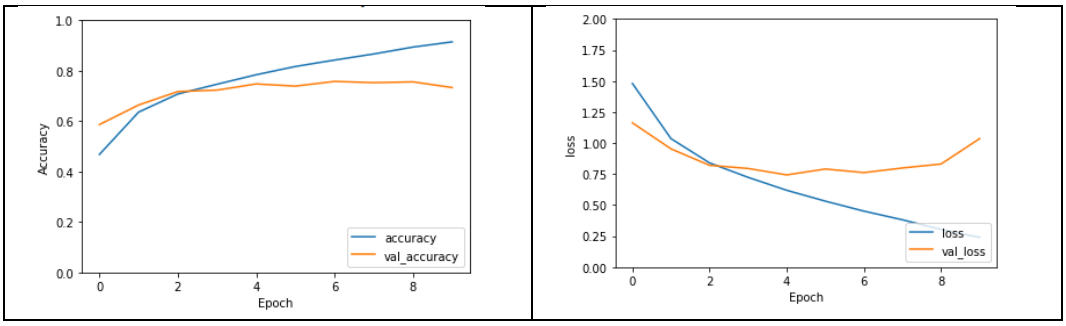
```
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

return cnn_model
```

Optimizer yang digunakan adalam **Adam optimizer** dengan learning **rate 0.001** dan losses yang digunakan **sparse_categorical_crossentropy**

Hasil yang didapatkan dalam proses training **10 epoch** dan batch **128**

loss: 0.2392 - accuracy: 0.9146 - val_loss: 1.0357 - val_accuracy: 0.7333

berikut grafik yang dihasilkan



Percobaan data tesing menggunakan model tersebut terhada random data test CIFAR10


```
Jumlah Prediksi benar: 9
Jumlah Prediksi Salah: 1
Rata-rata jawaban benar: 90.0 %
```

Percobaan selanjutnya untuk mengurang overfit pada val_loss dicoba dengan menambahkan **dropout 50%** pada dense layer

Hasil yang didapatkan dalam proses training **15 epoch** dan batch **128**

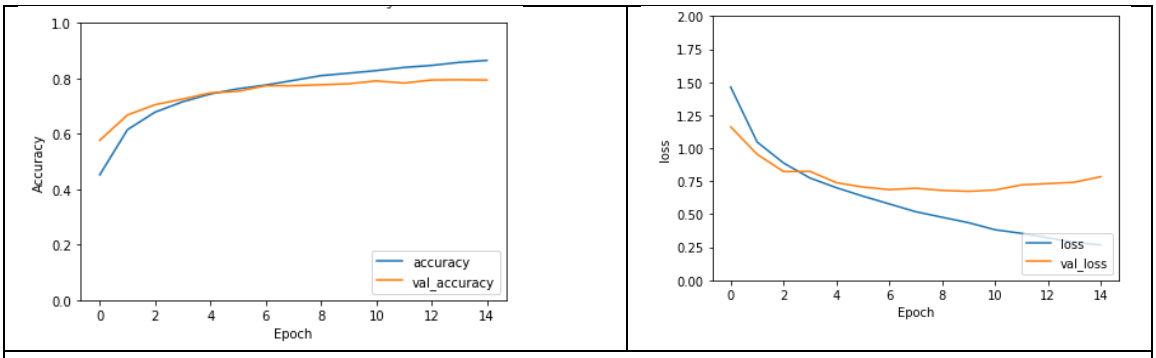
loss: 0.3804 - accuracy: 0.8651 - val_loss: 0.6252 - val_accuracy: 0.7943

Hasil yang didapatkan dalam proses training **50 epoch** dan batch **128**

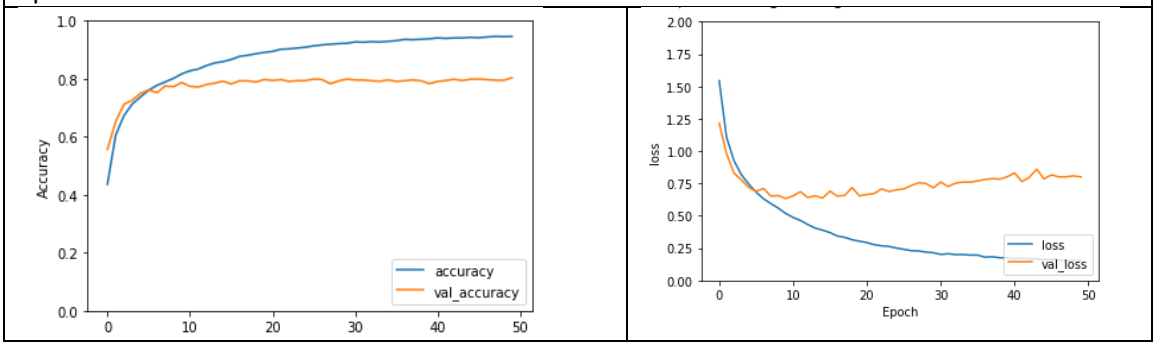
loss: 0.1521 - accuracy: 0.9459 - val_loss: 0.8007 - val_accuracy: 0.8038

Berikut grafik yang dihasilkan

Epoch 15



Epoch 50




```
Jumlah Prediksi benar: 8
Jumlah Prediksi Salah: 2
Rata-rata jawaban benar: 80.0 %
```

Berikut kesimpulan percobaan dari kedua arsitektur

No	Arsitektur	Epoch	Dropout	loss	accuracy	Val_loss	Val _accuracy
1	1	10	tidak	0.3754	0.87	0.91	0.72
2	1	10	20%	0.6139	0.78	0.79	0.72
3	2	10	tidak	0.239	0.915	1.035	0.73
4	2	50	tidak	0.041	0.987	2.47	0.726
5	2	15	50%	0.38	0.86	0.625	0.794
6	**2**	**50**	**50%**	**0.152**	**0.94**	**0.80**	**0.80**

Accuracy adalah kemampuan batch dalam mengekstraksi fitur, val_acc adalah kemampuan batch dalam mengingat fitur. Dropout dapat meningkatkan kemampuan batch dalam melakukan testing data tetapi dapat mengurangi kemampuan batch dalam melakukan training data.

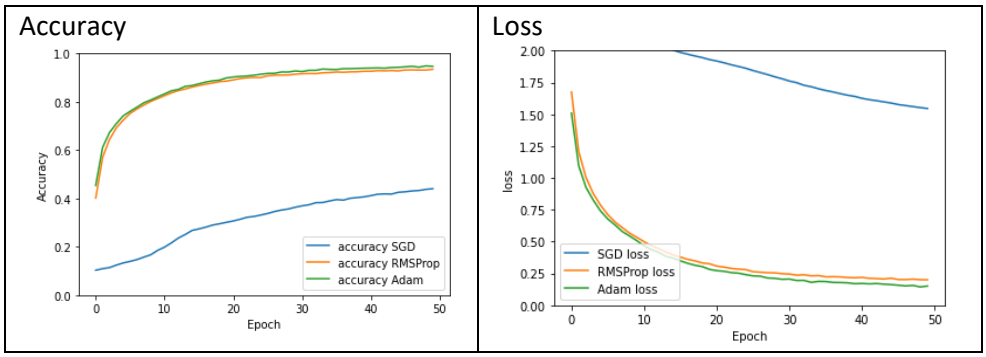
Berdasarkan percobaan diatas dapat disimpulkan bahwa arsitektur ke 2 dengan menambah dropout sebesar 50 persen memiliki kinerja paling baik (nomor urut 6).

Optimizer

Selanjutnya akan dilakukan percobaan optimizer dan losses menggunakan defenisi arsitektur nomor 6.

Dilakukan training dengan jumlah **epoch 50** untuk masing-masing Optimizer dengan **learning rate 0.001**

Optimizer	Loss	Accuracy	Val_loss	Val_accuracy
SGD	154	0.44	1.47	0.45
RMSProp	0.20	0.93	0.845	**0.80**
Adam	**0.15**	**0.94**	**0.81**	0.79



Optimizer adam memiliki kinerja paling baik dengan accuracy sebesar 94% dan loss 15%. Walaupun untuk val_accuracy RMSProp lebih baik sedikit dibandingkan adam.

Losses

Binary Cross entropy

loss: 9.9929 - accuracy: 0.1020 - val_loss: 9.9929 - val_accuracy: 0.1000

sparse_categorical_crossentropy

loss: 0.1521 - accuracy: 0.9459 - val_loss: 0.8007 - val_accuracy: 0.8038

D. Kesimpulan

Berdasarkan hasil eksporasi dapat disimpulkan bahwa :

1) Arsitektur CNN untuk data CIFAR menggunakan dasar arsitektur VGG dapat diimplementasikan dengan menambahkan Dropout sebesar 50%
2) Adam merupakan optimizer yang paling baik
3) Penerapan dropout cukup berpengaruh terhadap kinerka validation
4) Dapat dilakukan augmentasi data sebelum input untuk menangani noise atau kemiripan fitur.
5) Model yang dihasilkan dapat digunakan untuk testing data

2. EKSPLORASI REGRESI

A. Dataset

Dataset yang digunakan adalah Boston Housing Price yang diambil dari dataset *sklearn*, boston housing Price memiliki 13 variabel data dengan jumlah data sebanyak 506 data.

Percobaan Arsitektur 1

Menggunakan 1 hidden layer

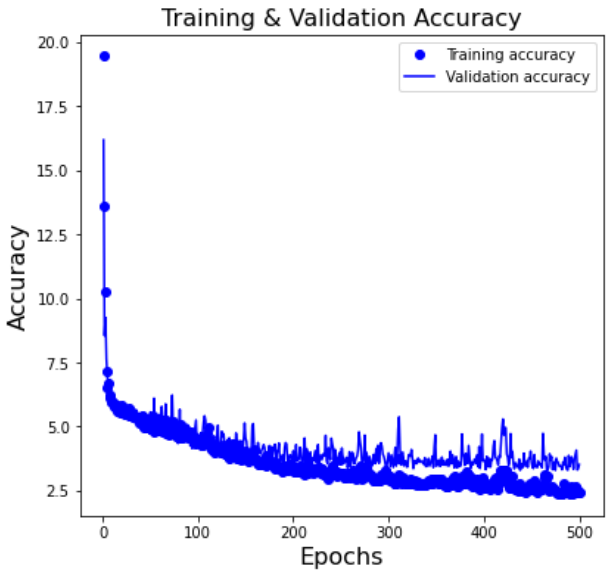
```
fc_model = tf.keras.Sequential([

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(13, activation=tf.nn.relu, input_shape=(13,)),

    tf.keras.layers.Dense(100, activation=tf.nn.relu),

    tf.keras.layers.Dense(1)


 ])

 return fc_model
```

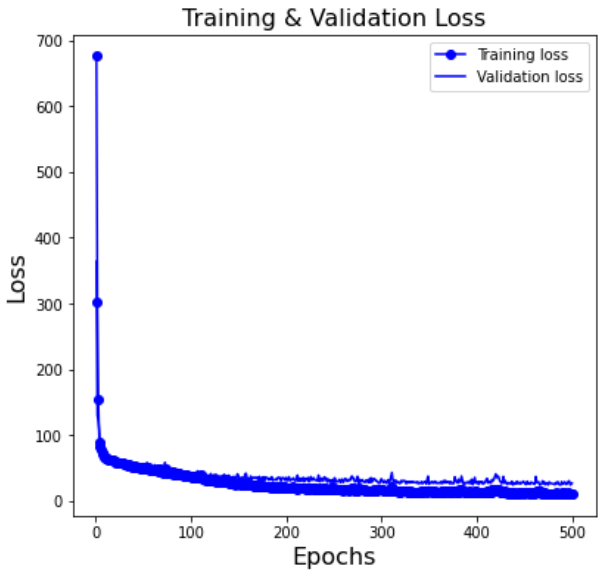
Optimizer yang digunakan adalam Adam optimizer dengan **learning rate 0.001** dan losses yang digunakan **mae** dan metrics **mse**

Hasil yang didapatkan dalam proses training **500 epoch** dan batch **32**

loss: 11.0103 - mae: 2.4051 - val_loss: 26.6783 - val_mae: 3.5158

berikut grafik yang dihasilkan





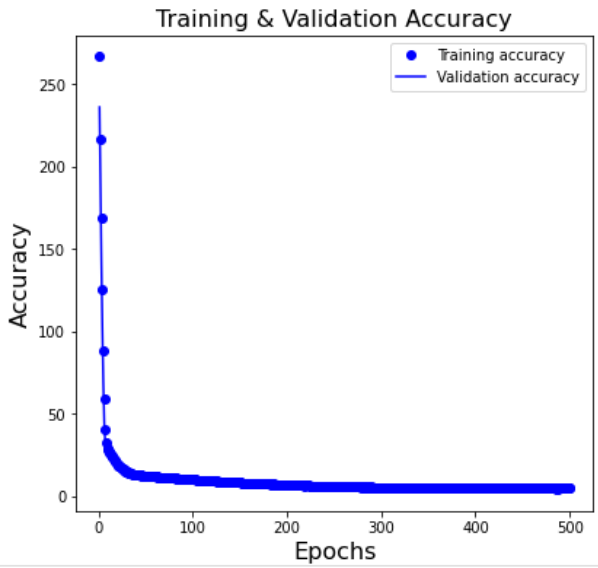
Percobaan Arsitektur 2

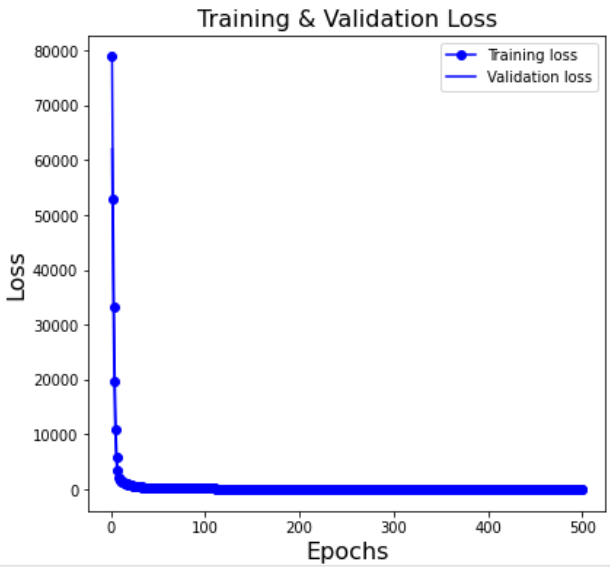
Tanpa hidden layer

```
def build_fc_model():
  fc_model = tf.keras.Sequential([
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(13, activation=tf.nn.relu, input_shape=(13,)),
      #tf.keras.layers.Dense(100, activation=tf.nn.relu),
      tf.keras.layers.Dense(1)
  ])
  return fc_model
```

Hasil yang didapatkan dalam proses training **500 epoch** dan batch **32**

loss: 29.6154 - mae: 4.1464 - val_loss: 38.4142 - val_mae: 4.0880





Percobaan Arsitektur 3

Dengan 2 hidden layer

```
fc_model = tf.keras.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(13, activation=tf.nn.relu, input_shape=(13,)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
```

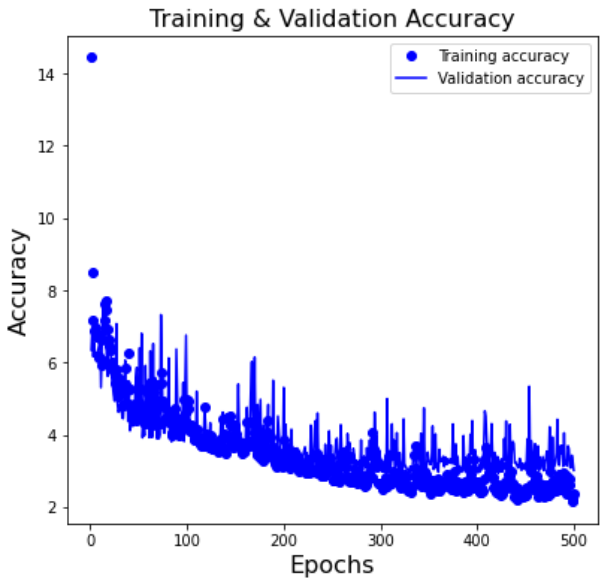
```
    tf.keras.layers.Dense(1)
])
return fc_model
```

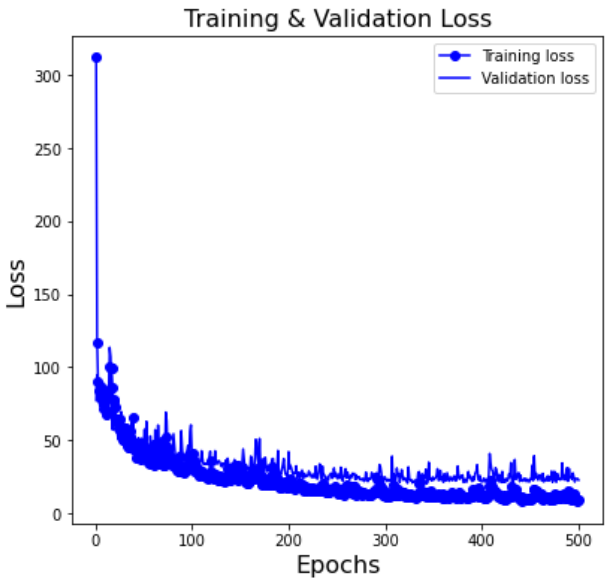
Hasil yang didapatkan dalam proses training **500 epoch** dan batch **32**

loss: 9.3781 - mae: 2.3783 - val_loss: 22.6837 - val_mae: 3.0142

berikut kesimpulan dari seluruh arsitektur

Arsitektur 1	Hidden Layer	Loss	MAE	Val Loss	Val MAE
1	1	11	2.4	26.6	3.5
2	0	29.6	4.14	38.4	4
3	**2**	**9.3**	**2.3**	**22.6**	**3.0**

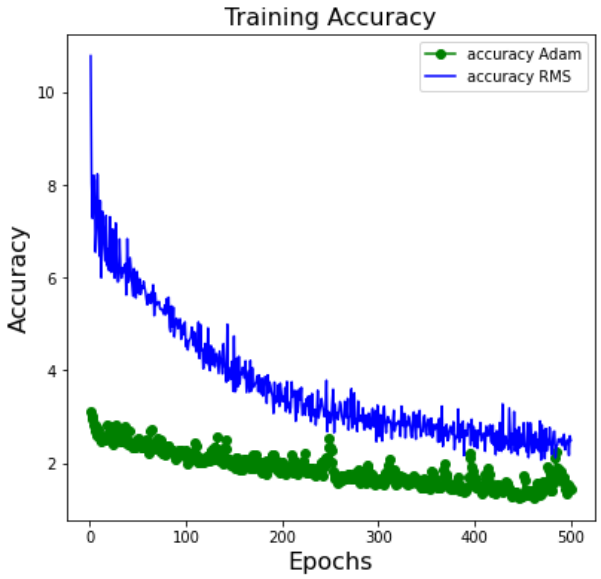


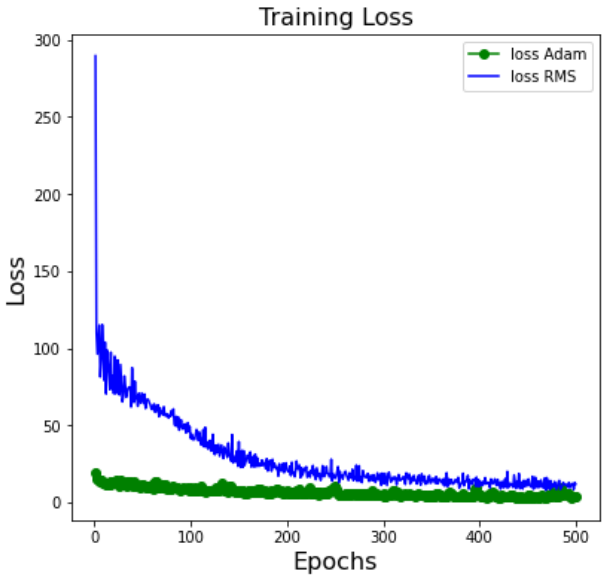


Selanjutnya akan dilakukan percobaan optimizer dan losses menggunakan defenisi arsitektur nomor 3.

Optimizer

Optimizer	Loss	MAE	Val Loss	Val MAE
Adam	**9.3**	**2.3**	**22.6**	**3.0**
RMSProp	11.7	2.48	30.54	3.7





Losses

Losses	Loss	MAE	Val Loss	Val MAE
mse	9.3	2.3	22.6	3.0
msle	9.8	76.19	9.7	75.1
huber	**1.89**	**2.33**	**2.56**	**3.00**

B. Kesimpulan

1) Jumlah hidden layer dapat mempengaruhi kinerja dalam pengenalan fitur
2) Adam merupakan Optimizer yang memiliki kinerja paling baik
3) Huber merupakan algoritma perhitungan loss yang memiliki kinerja paling baik dibanding dengan perhitungan loss lainnya