

## TUGAS BESAR

Nama : Freddy Artadima Silaban  
NIM : 33221046  
Mata Kuliah : Pembelajaran Mesin Lanjut

---

### 1. KLASIFIKASI

Untuk model CNN, lakukan eksplorasi untuk menjawab pertanyaan-pertanyaan sebagai berikut:

- Berapa banyaknya convolution layer yang optimal?
- Berapa ukuran filter yang optimal untuk setiap convolution layer?
- Berapa banyaknya filter yang optimal untuk setiap convolution layer?
- Berapa banyaknya hidden unit yang optimal pada bagian fully connected network?

Untuk mengetahui nilai yang paling optimal, harus dilakukan percobaan dengan membuat variasi nilai dari hyperparameter yang sedang dieksplorasi dengan nilai hyperparameter lainnya dibuat tetap (fixed). Jika ada nilai hyperparameter lainnya yang sudah ditemukan pada eksplorasi sebelumnya, gunakan nilai hyperparameter optimal tsb pada eksplorasi berikutnya. Nilai optimal diambil dari percobaan yang menghasilkan kinerja terbaik.

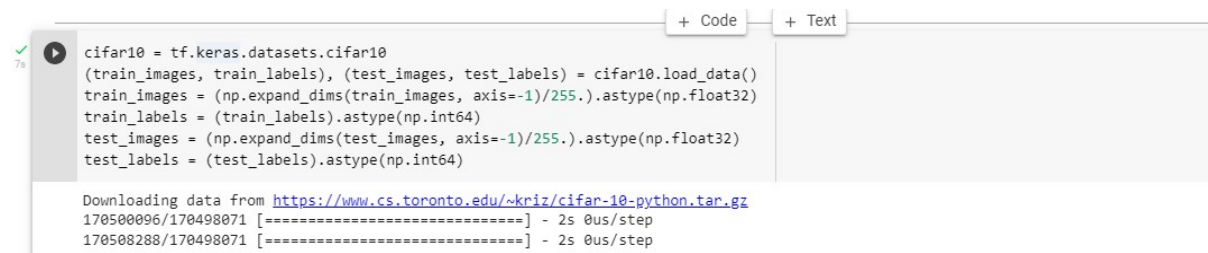
Selain itu, lakukan eksplorasi untuk menjawab pertanyaan berikut:

Dari semua pilihan yang disediakan oleh Keras *Optimizer*, mana yang menghasilkan kinerja paling baik (pada nilai parameter default)? Dari Keras *Optimizer* yang optimal (pada nilai parameter default), lakukan eksplorasi lebih lanjut apakah ada *learning rate schedule* yang menghasilkan kinerja yang lebih baik lagi. Dari semua pilihan yang disediakan oleh Keras (*Probabilistic*) *Losses*, mana yang menghasilkan kinerja paling baik?

#### Penyelesaian:

Klasifikasi adalah sebuah teknik untuk mengklasifikasikan atau mengkategorikan beberapa *item* yang belum berlabel ke dalam sebuah set kelas diskrit.

#### Pembacaan Data set:



```
cifar10 = tf.keras.datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
train_images = (np.expand_dims(train_images, axis=-1)/255.).astype(np.float32)
train_labels = (train_labels).astype(np.int64)
test_images = (np.expand_dims(test_images, axis=-1)/255.).astype(np.float32)
test_labels = (test_labels).astype(np.int64)
```

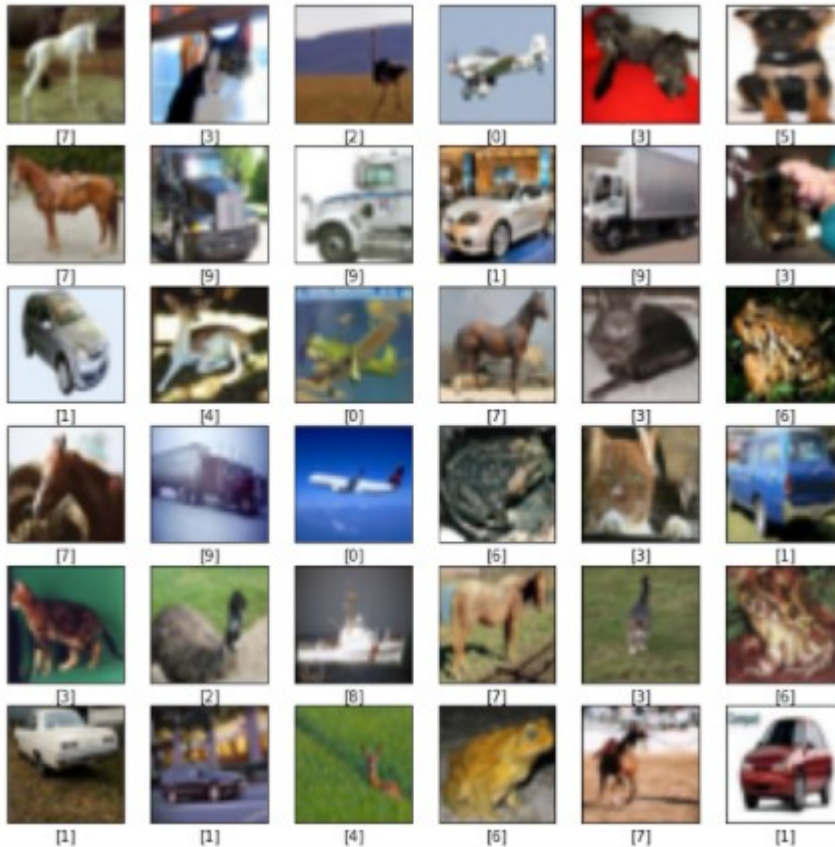
Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>  
170500096/170498071 [=====] - 2s 0us/step  
170508288/170498071 [=====] - 2s 0us/step

#### Tampilan Gambar Data Set:

```

plt.figure(figsize=(10,10))
random_inds = np.random.choice(50000,36)
for i in range(36):
    plt.subplot(6,6,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    image_ind = random_inds[i]
    plt.imshow(np.squeeze(train_images[image_ind]), cmap=plt.cm.binary)
    plt.xlabel(train_labels[image_ind])

```



## Train dan Model:

```

✓ [6] # Define the batch size and the number of epochs to use during training
18s BATCH_SIZE = 64
EPOCHS = 5

model.fit(train_images, train_labels, batch_size=BATCH_SIZE, epochs=EPOCHS)

Epoch 1/5
782/782 [=====] - 5s 4ms/step - loss: 2.0316 - accuracy: 0.2474
Epoch 2/5
782/782 [=====] - 3s 4ms/step - loss: 1.8868 - accuracy: 0.3056
Epoch 3/5
782/782 [=====] - 3s 4ms/step - loss: 1.8347 - accuracy: 0.3331
Epoch 4/5
782/782 [=====] - 3s 4ms/step - loss: 1.7921 - accuracy: 0.3534
Epoch 5/5
782/782 [=====] - 3s 4ms/step - loss: 1.7597 - accuracy: 0.3648
<keras.callbacks.History at 0x7fc401b12d10>

```

Proses langkah-langkah eksekusi yang dihasilkan terdiri dari 5 step dimana setiap step menghasilkan loss yang berbeda dan tingkat akurasi semakin baik. Tingkat akurasi semakin baik apabila nilai loss semakin sedikit, tingkat akurasi menurun apabila nilai loss semakin banyak.

### Pengujian akurasi dari Data Set

```
[ ] '''TODO: Use the evaluate method to test the model!'''
    test_loss, test_acc = model.evaluate(test_images, test_labels) # TODO
    # test_loss, test_acc = # TODO

    print('Test accuracy:', test_acc)

313/313 [=====] - 1s 3ms/step - loss: 1.8979 - accuracy: 0.3055
Test accuracy: 0.30550000071525574
```

Nilai tes akurasi yang dihasilkan sebesar: 0.3055

### Pertanyaan:

- A. Berapa banyaknya convolution layer yang optimal?
- B. Berapa ukuran filter yang optimal untuk setiap convolution layer?
- C. Berapa banyaknya filter yang optimal untuk setiap convolution layer?
- D. Berapa banyaknya hidden unit yang optimal pada bagian fully connected network?

### Jawaban:

Berdasarkan deskripsi hiperparameter di atas maka berikut deskripsi nilai hiperparameter yang akan dilakukan untuk mendapatkan akurasi model terbaik

- 1) CNN Layers = {3, 4}
- 2) Filter Count inside CNN = {24, 36}
- 3) Kernel Size = {3x3, 5x5}
- 4) Optimizer = {Adam, SGD, RMSprop}
- 5) Learning rate schedule = 0.023
- 6) Losses = {sparse\_categorical\_crossentropy, categorical\_crossentropy}
- 7) Initializer = {GlorotNormal, RandomNormal}
- 8) Dropout = {0.3, 0.7}

Nilai-nilai hyperparameter tersebut berkerja berdasarkan parameter default dari kasus digit recognition

BATCH\_SIZE = 64

EPOCHS = 5

## 2. REGRESI

### Persoalan Regresi

Gunakan arsitektur Fully Connected Neural Network dan dataset Boston Housing Price untuk persoalan regresi ini. Untuk model Fully Connected Neural

Network, lakukan eksplorasi untuk menjawab pertanyaan-pertanyaan sebagai berikut:

- a) Berapa banyaknya hidden layer yang optimal?
- b) Berapa banyaknya hidden unit yang optimal di setiap hidden layer?
- c) Apa activation function di setiap layer sehingga hasilnya optimal?
- d) Dari semua pilihan optimizer, apa optimizer yang hasilnya optimal? Dari semua pilihan loss function, apa yang hasilnya optimal?

### Penyelesaian:

Regresi adalah suatu teknik analisis untuk mengidentifikasi relasi atau hubungan diantara dua variabel atau lebih. Regresi bertujuan untuk menemukan suatu fungsi yang memodelkan data dengan meminimalkan *error* atau selisih antara nilai prediksi dengan nilai sebenarnya.

### Code Program:

```
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []

for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)
    model = build_model()
    model.fit(partial_train_data, partial_train_targets,
              epochs=num_epochs, batch_size=1, verbose=0)
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
```

### Penjelasan:

*Cross validation* adalah suatu metode tambahan dari teknik data mining yang bertujuan untuk memperoleh hasil akurasi yang maksimal.

Data.head dari Data Set:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Preprocessing:

```
y = data["MEDV"]
X = data.drop(labels = ["MEDV"],axis = 1)
train_data, test_data, train_targets, test_targets = train_test_split(X, y, test_size = 0.2, random_state=0)
```

```
#train_data.shape
#test_data.shape
train_targets
```

Model:

```
def build_model():
    model = models.Sequential()
    model.add(layers.Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

K-fold validation

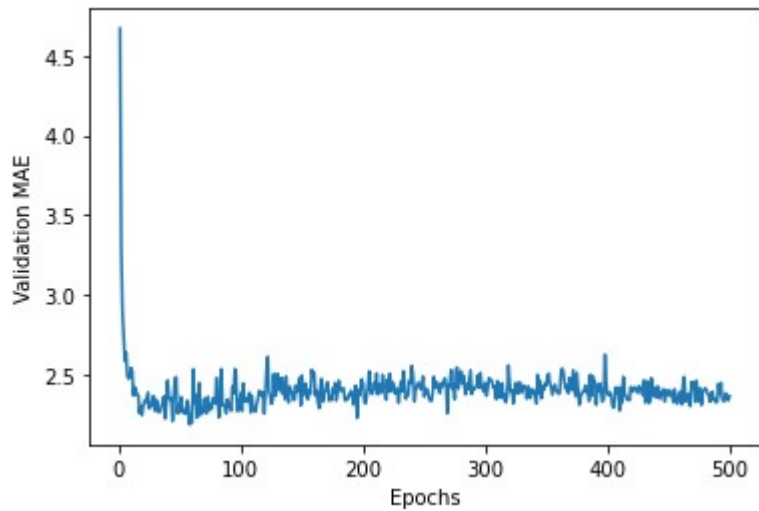
```
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []

for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)
    model = build_model()
    model.fit(partial_train_data, partial_train_targets,
              epochs=num_epochs, batch_size=1, verbose=0)
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
```

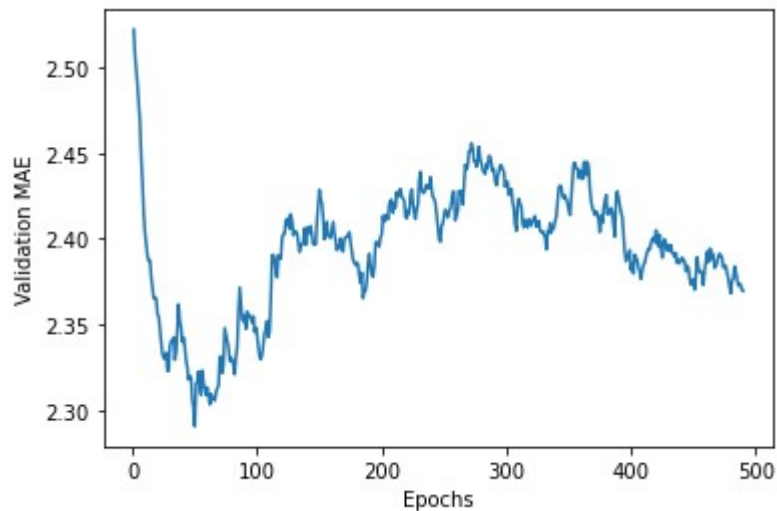


## Average MAE

```
plt.plot(range(1, len(average_mae_history) + 1), average_mae_history)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()
```



```
def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point * (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points
smooth_mae_history = smooth_curve(average_mae_history[10:])
plt.plot(range(1, len(smooth_mae_history) + 1), smooth_mae_history)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()
```



## Evaluate on test set

```
: model = build_model()
model.fit(train_data, train_targets,
epochs=80, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data, test_targets)
```

```
4/4 [=====] - 0s 2ms/step - loss: 19.4603 - mae: 2.9794
```

test\_mae\_score

2.9793686866760254

**Pertanyaan:**

- a) Berapa banyaknya hidden layar yang optimal?
- b) Berapa banyaknya hidden unit yang optimal di setiap hidden layar?
- c) Apa activation function di setiap layar sehingga hasilnya optimal?
- d) Dari semua pilihan optimizer, apa optimizer yang hasilnya optimal?
- e) Dari semua pilihan loss function, apa yang hasilnya optimal?

**Jawaban:**

Berdasarkan deskripsi hyperparameter di atas maka berikut deskripsi nilai hyperparameter yang akan dilakukan untuk mendapatkan akurasi model terbaik

- Jumlah Hidden Layers = {1, 2}
- Hidden Units = {13, 32, 64, 128}
- Activation function = {relu}
- Optimizer = {RMSprop}
- Losses = {Mean Squared Error (MSE)}

Nilai-nilai hyperparameter tersebut berkerja berdasarkan parameter default yaitu

- BATCH\_SIZE = 42
- EPOCHS = 500