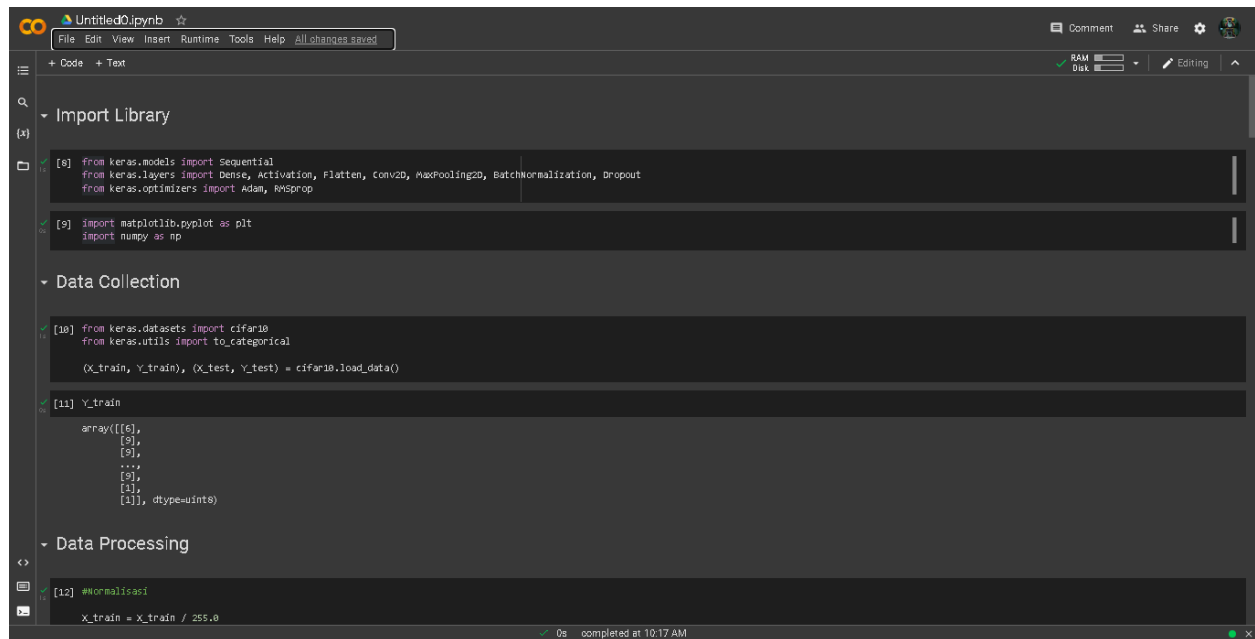


Ibrahim Bramullah

50420562 – 3IA02



The screenshot shows a Jupyter Notebook titled 'Untitled0.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for RAM, Disk, and Editing. The notebook is divided into sections: 'Import Library', 'Data Collection', and 'Data Processing'. The code in the 'Import Library' section imports Keras models, layers, optimizers, and Matplotlib. The 'Data Collection' section imports the CIFAR-10 dataset and loads it. The 'Data Processing' section shows the normalization of the training data.

```
[8] from keras.models import Sequential
    from keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D, BatchNormalization, Dropout
    from keras.optimizers import Adam, RMSprop

[9] import matplotlib.pyplot as plt
    import numpy as np

[10] from keras.datasets import cifar10
    from keras.utils import to_categorical
    (X_train, Y_train), (X_test, Y_test) = cifar10.load_data()

[11] Y_train
array([[6],
       [9],
       [9],
       ...,
       [9],
       [1],
       [1]], dtype=uint8)

[12] #normalisasi
    X_train = X_train / 255.0
```

1. import library
2. matplotlib = visualisasi  
numpy = perhitungan dengan angka
3. load data set
4. menampilkan Y\_train

```
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Data Processing
[12] #normalisasi
X_train = X_train / 255.0
X_test = X_test / 255.0

[14] X_train
array([[[[0.23137255, 0.24313725, 0.24705882],
         [0.15862745, 0.18039216, 0.17647059],
         [0.19607843, 0.18823529, 0.16862745],
         ...,
         [0.61960784, 0.51764706, 0.42352941],
         [0.59607843, 0.43019608, 0.4       ],
         [0.58039216, 0.48627451, 0.40392157]],
        [[0.0627451, 0.07843137, 0.07843137],
         [0.       , 0.       , 0.       ],
         [0.07058824, 0.03137255, 0.       ],
         ...,
         [0.48235294, 0.34509804, 0.21568627],
         [0.46666667, 0.3254902, 0.19607843],
         [0.47943137, 0.34117647, 0.22352941]],
        [[0.09803922, 0.09411765, 0.08235294],
         [0.0627451, 0.02745098, 0.       ],
         [0.19215686, 0.10588235, 0.03137255],
         ...,
         [0.4627451, 0.32941176, 0.19607843],
         [0.47058824, 0.32941176, 0.19607843],
         [0.42745098, 0.28627451, 0.16470588]],
        ...,
        [[0.81568627, 0.66666667, 0.37647059],
         [0.78823529, 0.6       , 0.13333333],
         [0.77647059, 0.63137255, 0.10196078],
         ...,
         [0.62745098, 0.52156863, 0.2745098 ],
         [0.21960784, 0.12156863, 0.02745098],
         [0.20784314, 0.13333333, 0.07843137]]]])

0s completed at 10:17 AM
```

5. normalisasi angka (dari 0 sampai 1)

6. menampilkan x\_train

```
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[14]
[[[0.70588235, 0.54509804, 0.37647059],
  [0.67943137, 0.48235294, 0.16470588],
  [0.72941176, 0.56470588, 0.11764706],
  ...,
  [0.72156863, 0.58039216, 0.36862745],
  [0.58039216, 0.24313725, 0.13333333],
  [0.3254902, 0.20784314, 0.13333333]],
 [[0.69411765, 0.56470588, 0.4540126],
  [0.65882353, 0.50588235, 0.36862745],
  [0.70196078, 0.55686275, 0.34117647],
  ...,
  [0.84705882, 0.72156863, 0.54901961],
  [0.59215686, 0.4627451, 0.32941176],
  [0.48235294, 0.36078431, 0.28235294]]],
 [[0.60392157, 0.69411765, 0.73333333],
  [0.49411765, 0.5372549, 0.53333333],
  [0.41176471, 0.40784314, 0.37254902],
  ...,
  [0.35686275, 0.37254902, 0.27943137],
  [0.34117647, 0.35294118, 0.27943137],
  [0.30980392, 0.31764706, 0.2745098 ]]],
 ...,
 [[0.60392157, 0.69411765, 0.73333333],
  [0.49411765, 0.5372549, 0.53333333],
  [0.41176471, 0.40784314, 0.37254902],
  ...,
  [0.35686275, 0.37254902, 0.27943137],
  [0.34117647, 0.35294118, 0.27943137],
  [0.30980392, 0.31764706, 0.2745098 ]]]]

[15] from keras.utils import to_categorical
Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)

[16] Y_train
array([[0., 0., ..., 0., 0., 0.],
       [0., 0., ..., 0., 0., 1.],
       [0., 0., ..., 0., 0., 1.],
       ...,
       [0., 0., ..., 0., 0., 1.],
       [0., 1., ..., 0., 0., 0.],
       [0., 1., ..., 0., 0., 0.]], dtype=float32)

Modeling
0s completed at 10:17 AM
```

7. membuat Y/X\_train menjadi kategorikal

8. menampilkan Y\_train

The screenshot shows a Jupyter Notebook with a Python code cell defining a CNN model. The code starts with `input_shape = (32,32,3)` and builds a `Sequential` model with layers: `conv2D(16, (3,3), padding='same', input_shape = input_shape)`, `Activation('relu')`, `conv2D(16, (3,3))`, `Activation('relu')`, `MaxPooling2D((2,2), padding='same')`, `Dropout(0.4)`, `conv2D(32, (3,3))`, `Activation('relu')`, `MaxPooling2D((2,2), padding='same')`, `Dropout(0.4)`, `conv2D(64, (3,3))`, `Activation('relu')`, `MaxPooling2D((2,2), padding='same')`, `Dropout(0.4)`, `conv2D(128, (3,3))`, `Activation('relu')`, `MaxPooling2D((2,2), padding='same')`, `Dropout(0.4)`, `Flatten()`, `Dense(1000)`, `Activation('relu')`, `Dropout(0.4)`, and `Dense(10, activation='softmax')`. A second cell runs `model_cnn.summary()`, displaying a table of the model's layers.

Layer (type)	Output Shape	Param #
conv2d (conv2d)	(None, 32, 32, 16)	448

9. memanggil shape dan menyamakan dengan ukuran shape yang diinginkan  
juga tidak lupa merubah angka dari gambar menjadi vektor dengan flatten

This screenshot shows the same Jupyter Notebook after adding a `model_cnn.add(Dense(10), activation='softmax')` layer. The `model_cnn.summary()` cell now displays a more complete table of the model's layers.

Layer (type)	Output Shape	Param #
conv2d (conv2d)	(None, 32, 32, 16)	448
activation (Activation)	(None, 32, 32, 16)	0
conv2d_1 (conv2d)	(None, 30, 30, 16)	2320
activation_1 (Activation)	(None, 30, 30, 16)	0
max_pooling2d (MaxPooling2D)	(None, 15, 15, 16)	0
dropout (Dropout)	(None, 15, 15, 16)	0
conv2d_2 (conv2d)	(None, 13, 13, 32)	4640
activation_2 (Activation)	(None, 13, 13, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 32)	0
dropout_1 (Dropout)	(None, 7, 7, 32)	0
conv2d_3 (conv2d)	(None, 5, 5, 64)	18496
activation_3 (Activation)	(None, 5, 5, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 64)	0
dropout_2 (Dropout)	(None, 3, 3, 64)	0
conv2d_4 (conv2d)	(None, 1, 1, 128)	73856
activation_4 (Activation)	(None, 1, 1, 128)	0
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 128)	0
dropout_3 (Dropout)	(None, 1, 1, 128)	0

10. menampilkan proses diatas/sblmnya

```

[22] model_cnn.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])

[24] hist = model_cnn.fit(X_train, Y_train,
                        epochs=25,
                        batch_size=300,
                        shuffle=True,
                        validation_data=(X_test, Y_test))

Epoch 1/25
250/250 [=====] - 96s 378ms/step - loss: 1.9422 - accuracy: 0.2617 - val_loss: 1.6474 - val_accuracy: 0.4074
Epoch 2/25
250/250 [=====] - 92s 366ms/step - loss: 1.5987 - accuracy: 0.4086 - val_loss: 1.5457 - val_accuracy: 0.4381
Epoch 3/25
250/250 [=====] - 91s 364ms/step - loss: 1.4670 - accuracy: 0.4615 - val_loss: 1.3409 - val_accuracy: 0.5140
Epoch 4/25
250/250 [=====] - 91s 364ms/step - loss: 1.3987 - accuracy: 0.4892 - val_loss: 1.3144 - val_accuracy: 0.5331
Epoch 5/25
250/250 [=====] - 91s 363ms/step - loss: 1.3457 - accuracy: 0.5109 - val_loss: 1.2259 - val_accuracy: 0.5612
Epoch 6/25
250/250 [=====] - 91s 363ms/step - loss: 1.2929 - accuracy: 0.5288 - val_loss: 1.1458 - val_accuracy: 0.5930
Epoch 7/25
250/250 [=====] - 91s 363ms/step - loss: 1.2603 - accuracy: 0.5445 - val_loss: 1.0988 - val_accuracy: 0.6072
Epoch 8/25
250/250 [=====] - 90s 362ms/step - loss: 1.2307 - accuracy: 0.5544 - val_loss: 1.0748 - val_accuracy: 0.6176
Epoch 9/25
250/250 [=====] - 91s 363ms/step - loss: 1.1988 - accuracy: 0.5690 - val_loss: 1.0527 - val_accuracy: 0.6193
Epoch 10/25
250/250 [=====] - 90s 362ms/step - loss: 1.1719 - accuracy: 0.5802 - val_loss: 1.0456 - val_accuracy: 0.6322
Epoch 11/25
250/250 [=====] - 92s 367ms/step - loss: 1.1554 - accuracy: 0.5978 - val_loss: 0.9959 - val_accuracy: 0.6512
Epoch 12/25
250/250 [=====] - 91s 363ms/step - loss: 1.1345 - accuracy: 0.5954 - val_loss: 0.9602 - val_accuracy: 0.6662
Epoch 13/25
250/250 [=====] - 91s 362ms/step - loss: 1.1209 - accuracy: 0.5982 - val_loss: 0.9837 - val_accuracy: 0.6558
Epoch 14/25
250/250 [=====] - 90s 359ms/step - loss: 1.0997 - accuracy: 0.6090 - val_loss: 0.9510 - val_accuracy: 0.6617
Epoch 15/25
250/250 [=====] - 90s 358ms/step - loss: 1.0937 - accuracy: 0.6082 - val_loss: 0.9345 - val_accuracy: 0.6737
Epoch 16/25
250/250 [=====] - 90s 359ms/step - loss: 1.0849 - accuracy: 0.6124 - val_loss: 0.9238 - val_accuracy: 0.6735
0s completed at 10:17 AM

```

- 11. mengkompilke menggunakan 'adam'
- 12. melatih modeling agar tidak sesuai index

```

[26] score = model_cnn.evaluate(X_test, Y_test)
print('Accuracy: ', score[1]*100, "%")

313/313 [=====] - 9s 298ms/step - loss: 0.8586 - accuracy: 0.6990
Accuracy: 69.90000009536743 %

[27] #show Accuracy Graph
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.legend(['acc', 'val_acc'])
plt.show()

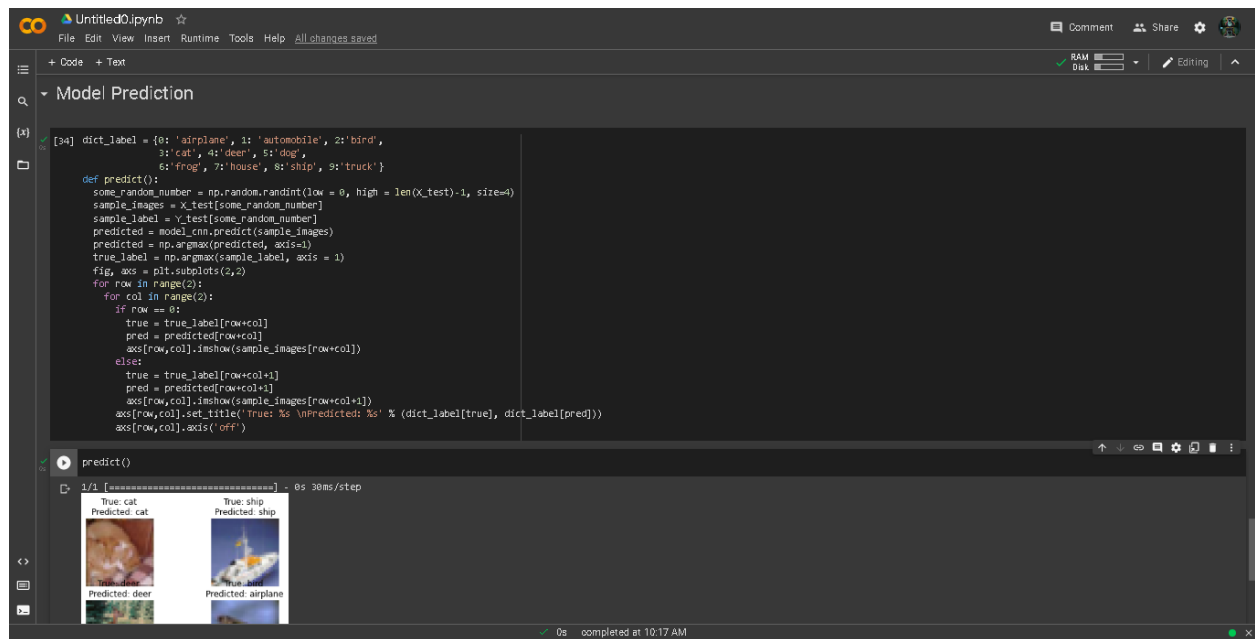
Model Evaluation
Model Prediction

[34] dict_label = {0: 'airplane', 1: 'automobile', 2: 'bird',
                3: 'cat', 4: 'deer', 5: 'dog',
                6: 'frog', 7: 'house', 8: 'ship', 9: 'truck'}

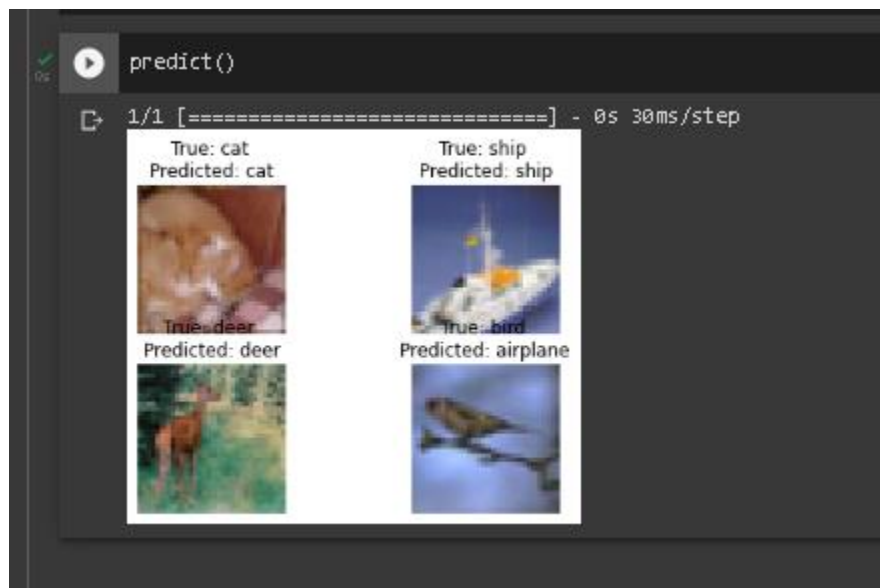
def predict():
    some_random_number = np.random.randint(low=0, high=len(X_test)-1, size=4)
    sample_images = X_test[some_random_number]

```

- 13. menampilkan akurasi model
- 14. menampilkan akurasi dengan graph
- 15. menampilkan akurasi error value



16. prediksi model yang sudah diuji  
(berbeda prediksi di akhir pengujian)



17. menampilkan prediksi