

**LAPORAN PRAKTIKUM
KECERDASAN BUATAN**

MODUL 10

MACHINE EARNING : KLASIFIKASI SINYAL ELEKTROMEDIK

DISUSUN OLEH :

SANDRO ANUGRAH TAMBUNAN 2250081136



**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN INFORMATIKA
UNIVERSITAS JENDERAL ACHMAD YANI
TAHUN 2024**

DAFTAR ISI

BAB I.	HASIL PRAKTIKUM.....	2
I.1	Program I.....	2
I.1.A.	Langkah Kerja/Source Code	2
I.1.B.	Screenshot	5
I.1.C.	Analisis.....	6
BAB II.	KESIMPULAN.....	8
BAB III.	LINK GOOGLE COLAB	9

BAB I. HASIL PRAKTIKUM

I.1 Program I

I.1.A. Langkah Kerja/Source Code

```
import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.model_selection import train_test_split

import tensorflow as tf

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.utils import to_categorical


# LOAD DATA

data = pd.read_csv('D:/PRAKTIKUM KECERDASAN BUATAN/MODUL
10/emotions.csv')


# TRANSFORM LABEL TO NUMBER

def Transform_data(data):

    encoding_data = {'NEUTRAL': 0, 'POSITIVE': 1,
'NEGATIVE': 2}

    data_encoded = data.replace(encoding_data)

    x = data_encoded.drop(["label"], axis=1)

    y = data_encoded.loc[:, 'label'].values

    scaler = StandardScaler()
```

```

        scaler.fit(x)

        X = scaler.transform(x)

        Y = to_categorical(y)

        return X, Y

X, Y = Transform_data(data)

x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=4)

# CREATE MODEL WITH LSTM
def create_model():

    inputs = tf.keras.Input(shape=(x_train.shape[1],))

    reshape = tf.keras.layers.Reshape((x_train.shape[1],
1))(inputs)

    lstm = tf.keras.layers.LSTM(256,
return_sequences=True)(reshape)

    flatten = tf.keras.layers.Flatten()(lstm)

    outputs = tf.keras.layers.Dense(3,
activation='softmax')(flatten)

    model = tf.keras.Model(inputs=inputs,
outputs=outputs)

    print(model.summary())

    return model

```

```
lsmtmodel = create_model()

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

# LEARNING PROCESS

history = lsmtmodel.fit(x_train, y_train, epochs=10,
validation_split=0.1)

loss, acc = lsmtmodel.evaluate(x_test, y_test)

print(f"Loss On Testing: {loss*100}", f"\nAccuracy On
Training: {acc*100}")

# PREDICTION

pred = lsmtmodel.predict(x_test)
pred1 = np.argmax(pred, axis=1)
y_test1 = np.argmax(y_test, axis=1)
```

I.1.B. Screenshot

The first screenshot shows the initial data loading and transformation steps in a Jupyter Notebook. The code imports necessary libraries (numpy, pandas, seaborn, sklearn, tensorflow, keras) and loads the 'emotions.csv' dataset. It then displays the value counts for the 'label' column, showing a distribution of 716 Neutral, 708 Negative, and 708 Positive instances. A transformation function is defined to encode the labels and scale the features.

```
[1] import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import to_categorical

data = pd.read_csv('/content/emotions.csv')

data['label'].value_counts()

label
NEUTRAL    716
NEGATIVE   708
POSITIVE   708
Name: count, dtype: int64

def Transform_data(data):
    encoding_data = {'NEUTRAL': 0, 'POSITIVE': 1, 'NEGATIVE': 2}
    data_encoded = data.replace(encoding_data)
    x = data_encoded.drop(['label'], axis=1)
    y = data_encoded['label'].values
    if not np.all(np.isin(y, list(encoding_data.values()))):
        raise ValueError("Ada nilai label yang tidak sesuai dengan encoding data.")
    scaler = StandardScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    y = to_categorical(y)
```

The second screenshot shows the model compilation and training process. The 'lstm_model' is created with 'adam' as the optimizer and 'categorical_crossentropy' as the loss function. The model's architecture is displayed, showing an input layer, a reshape layer, an LSTM layer, a flatten layer, and a dense output layer. The training process is initiated using 'lstm_model.fit'.

```
lstm_model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

Model: "lstm_model"
Layer (type)                 Output Shape          Param #
-----
input_1 (InputLayer)         [(None, 2548)]        0
reshape (Reshape)            (None, 2548, 1)       0
lstm (LSTM)                  (None, 2548, 256)     264192
flatten (Flatten)            (None, 652288)        0
dense (Dense)                (None, 3)             1956867
Total params: 2221059 (8.47 MB)
Trainable params: 2221059 (8.47 MB)
Non-trainable params: 0 (0.00 Byte)

None

history = lstm_model.fit(x_train, y_train, epochs=10, validation_split=0.1)
loss, acc = lstm_model.evaluate(x_test, y_test)
```

The third screenshot shows the training progress and evaluation results. The training history is displayed, showing the loss and accuracy for each epoch. The final loss and accuracy are printed, along with the predicted values for the test set.

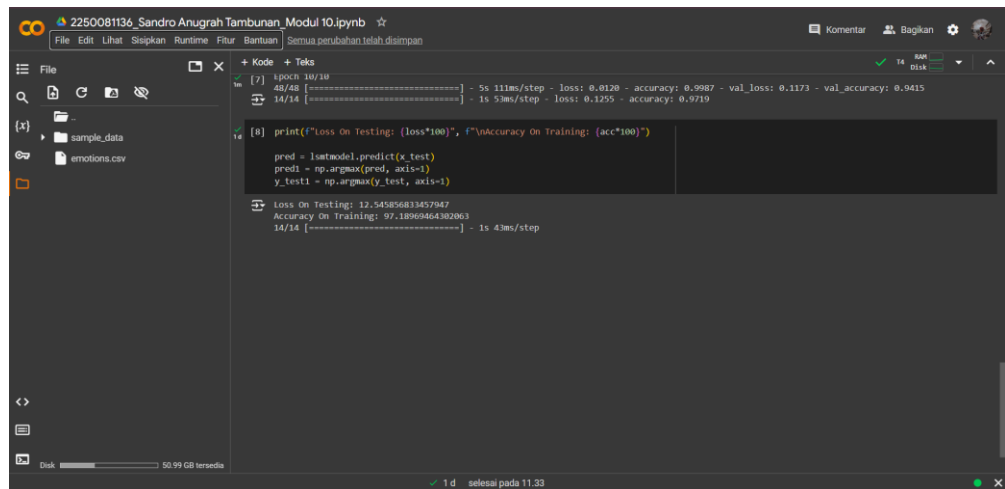
```
history = lstm_model.fit(x_train, y_train, epochs=10, validation_split=0.1)
loss, acc = lstm_model.evaluate(x_test, y_test)

Epoch 1/10
40/40 [-----] - 16s 192ms/step - loss: 0.3827 - accuracy: 0.8729 - val_loss: 0.3403 - val_accuracy: 0.8947
Epoch 2/10
40/40 [-----] - 5s 114ms/step - loss: 0.2138 - accuracy: 0.9179 - val_loss: 0.3920 - val_accuracy: 0.9415
Epoch 3/10
40/40 [-----] - 5s 107ms/step - loss: 0.1845 - accuracy: 0.9302 - val_loss: 0.1779 - val_accuracy: 0.9298
Epoch 4/10
40/40 [-----] - 5s 110ms/step - loss: 0.0859 - accuracy: 0.9687 - val_loss: 0.1765 - val_accuracy: 0.9298
Epoch 5/10
40/40 [-----] - 5s 111ms/step - loss: 0.1282 - accuracy: 0.9563 - val_loss: 0.4912 - val_accuracy: 0.9649
Epoch 6/10
40/40 [-----] - 5s 108ms/step - loss: 0.0802 - accuracy: 0.9674 - val_loss: 0.1355 - val_accuracy: 0.9474
Epoch 7/10
40/40 [-----] - 5s 114ms/step - loss: 0.0547 - accuracy: 0.9817 - val_loss: 0.1586 - val_accuracy: 0.9474
Epoch 8/10
40/40 [-----] - 5s 108ms/step - loss: 0.0333 - accuracy: 0.9896 - val_loss: 0.1533 - val_accuracy: 0.9532
Epoch 9/10
40/40 [-----] - 5s 112ms/step - loss: 0.0260 - accuracy: 0.9928 - val_loss: 0.1029 - val_accuracy: 0.9532
Epoch 10/10
40/40 [-----] - 5s 111ms/step - loss: 0.0120 - accuracy: 0.9987 - val_loss: 0.1173 - val_accuracy: 0.9415
14/14 [-----] - 1s 53ms/step - loss: 0.1255 - accuracy: 0.9719

[8] print(f"loss On Testing: {loss*100}", f"Accuracy On Training: {acc*100}")

pred = lstm_model.predict(x_test)
pred1 = np.argmax(pred, axis=1)
y_test1 = np.argmax(y_test, axis=1)

loss On Training: 13.4365622267047
```



```
File Edit Lihat Siapkan Runtime Fitur Bantuan Semua perubahan telah disimpan
+ Kode + Teks
epoch 10/10
48/48 [ ] - 5s 111ms/step - loss: 0.0120 - accuracy: 0.9987 - val_loss: 0.1173 - val_accuracy: 0.9415
14/14 [ ] - 1s 53ms/step - loss: 0.1255 - accuracy: 0.9719

[8] print(f"Loss On Testing: {loss*100}", f"Accuracy On Training: {acc*100}")

pred = lastmodel.predict(y_test)
predi = np.argmax(pred, axis=1)
y_testi = np.argmax(y_test, axis=1)

Loss On Testing: 12.545806813457047
Accuracy On Training: 97.18060464302863
14/14 [ ] - 1s 43ms/step
```

I.1.C. Analisis

Dari kode diatas setelah memuat dataset emosi, script yang diberikan diproses untuk klasifikasi menggunakan model LSTM dalam TensorFlow. Label kategori ("NEUTRAL", "POSITIVE", "NEGATIVE") diubah menjadi nilai numerik (0, 1, 2). Selain itu, fitur-fitur diskalakan dan label-label diencode menggunakan encoding satu panas. Selanjutnya, data dibagi menjadi set pelatihan dan pengujian.

Model LSTM terdiri dari lapisan input yang mengubah bentuk data untuk memenuhi persyaratan lapisan LSTM. Lapisan berikutnya terdiri dari lapisan LSTM dengan 256 unit, lapisan flatten, dan lapisan output tebal dengan aktivasi softmax untuk klasifikasi multiclass. Model ini dikompilasi menggunakan pengurangan cross-entropy kategoris dan optimizer Adam.

Selama fase pelatihan, model dilatih pada set pelatihan selama sepuluh epoch dengan split validasi sebesar sepuluh persen. Setelah fase pelatihan selesai, kinerja model dievaluasi pada set pengujian, yang menghasilkan nilai kehilangan dan akurasi tes. Prediksi dibuat pada set pengujian, dan label sebenarnya dan prediksi diubah kembali ke format kategori untuk analisis dari format one-hot encoded.

Secara keseluruhan, script ini berhasil mengimplementasikan pipeline pembelajaran mendalam untuk klasifikasi emosi, mulai dari pemrosesan data hingga evaluasi model. Namun, untuk meningkatkan kinerjanya, script ini dapat memperoleh manfaat dari optimasi dan penyetelan hyperparameter tambahan, seperti jumlah unit LSTM, epoch, dan laju pembelajaran. Menggabungkan teknik seperti lapisan dropout juga dapat membantu meningkatkan generalisasi model dan mengurangi overfitting.

BAB II. KESIMPULAN

Pada pertemuan praktikum modul 10 ini, saya mempelajari menggunakan model LSTM dalam TensorFlow untuk mempelajari bagaimana menggunakan pipeline pembelajaran mendalam untuk klasifikasi emosi. Proses ini mencakup pemuatan dan pemrosesan dataset emosi; selain itu, label kategori diubah menjadi nilai numerik, dan elemen data diskalakan dan diencode menggunakan teknik one-hot encoding. Untuk mempersiapkan model LSTM, data kemudian dibagi menjadi set pengujian dan pelatihan.

Model LSTM terdiri dari beberapa lapisan untuk klasifikasi multiclass; ini termasuk lapisan input yang mengubah bentuk data, lapisan LSTM dengan 256 unit, lapisan flatten, dan lapisan output yang diaktifkan softmax. Model ini dikompilasi dengan menggunakan optimizer Adam dan loss categorical cross-entropy. Selama fase pelatihan, model dilatih pada set pelatihan dengan validasi 10%. Hasil evaluasi menunjukkan nilai kehilangan dan akurasi set pengujian.

Setelah itu, label diubah dari satu-hot encoded format ke format kategori asli untuk memeriksa hasil prediksi. Secara keseluruhan, penggunaan ini berhasil; namun, melalui optimasi dan penyetelan hyperparameter tambahan, kinerja model masih dapat ditingkatkan. Teknik seperti lapisan dropout juga dapat membantu meningkatkan generalisasi model dan mengurangi overfitting. Hasilnya, praktikum ini memberikan pemahaman yang lebih baik tentang penggunaan LSTM dalam klasifikasi emosi. Selain itu, tahap-tahap pemrosesan data dan penyetelan model sangat penting untuk mendapatkan hasil yang optimal.

BAB III. LINK GOOGLE COLAB

<https://colab.research.google.com/drive/1JdDprvKSk-yOhra2BubCbbnVl331r89v?usp=sharing>