**Comparison of LSTM, BiLSTM, GRU, BiGRU, and CNN Models**

Classification Analysis of Sentiment on Tweets Regarding President Putin's Participation in Indonesia's G20 with the BiGRU Method

# **LSTM**

1. LSTM Model

def model\_lstm(num\_words, embedding\_dim, maxlen):

    tf.random.set\_seed(123)

    lstm\_dim = 32

    dense\_dim = 6

    modelLSTM = tf.keras.Sequential([

        tf.keras.layers.Embedding(num\_words, embedding\_dim, input\_length=maxlen),

        tf.keras.layers.LSTM(lstm\_dim),

        tf.keras.layers.Dense(dense\_dim, activation='relu'),

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

    ])

    modelLSTM.compile(loss='sparse\_categorical\_crossentropy',

                  optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

                  metrics=['accuracy'])

    return modelLSTM

modelLSTM = model\_lstm(NUM\_WORDS, EMBEDDING\_DIM, MAXLEN)

modelLSTM.summary()

1. Model.fit

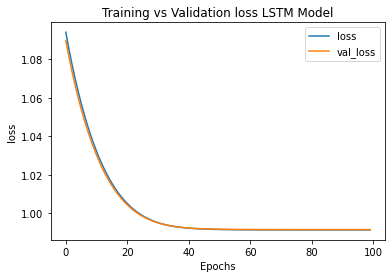
historyLSTM = modelLSTM.fit(train\_padded\_seq,

                            train\_label\_seq,

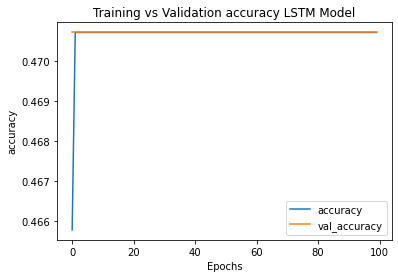
                            epochs=100,

                            validation\_data=(val\_padded\_seq, val\_label\_seq))

1. Results



*Training and Validation Loss for LSTM Model*

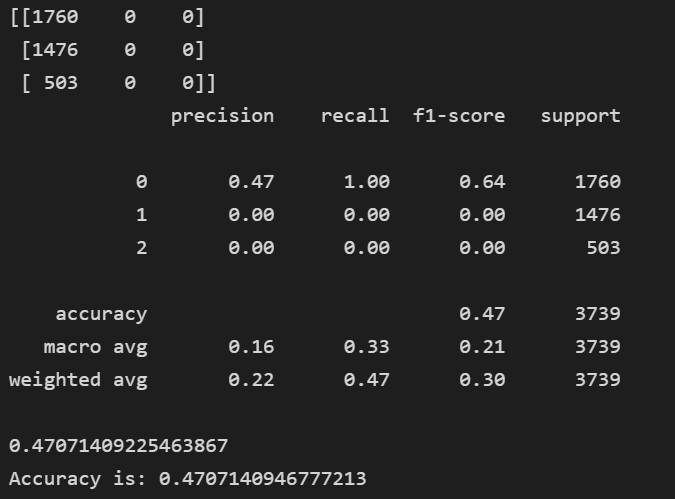
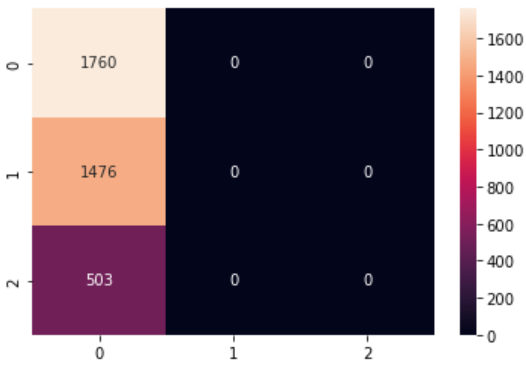


*Training and Validation Accuracy for LSTM Model*

Best Validation Accuracy : 0.4707

On Epoch : 1

Confusin Matrix :

# BiLSTM

1. BiLSTM Model

def model\_BiLSTM(num\_words, embedding\_dim, maxlen):

    tf.random.set\_seed(123)

    lstm\_dim = 32

    dense\_dim = 6

    modelBiLSTM = tf.keras.Sequential([

        tf.keras.layers.Embedding(num\_words, embedding\_dim, input\_length=maxlen),

        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(lstm\_dim)),

        tf.keras.layers.Dense(dense\_dim, activation='relu'),

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

    ])

    modelBiLSTM.compile(loss='sparse\_categorical\_crossentropy',

                  optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

                  metrics=['accuracy'])

    return modelBiLSTM

modelBiLSTM = model\_BiLSTM(NUM\_WORDS, EMBEDDING\_DIM, MAXLEN)

modelBiLSTM.summary()

1. ModelCheckpoint & Model.fit

mc = ModelCheckpoint('classifier-putinG20-BiLSTM\_01.h5', monitor='val\_accuracy', mode='max', verbose=1, save\_best\_only=True)

historyBiLSTM = modelBiLSTM.fit(train\_padded\_seq,

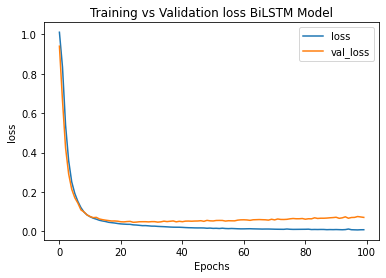
                                train\_label\_seq,

                                epochs=100,

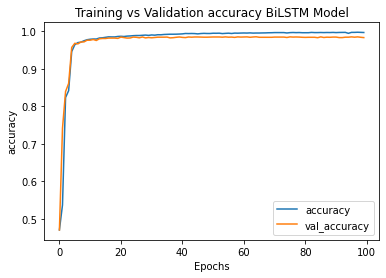
                                callbacks=[mc],

                                validation\_data=(val\_padded\_seq, val\_label\_seq))

1. Result



*Training and Validation Loss for BiLSTM Model*

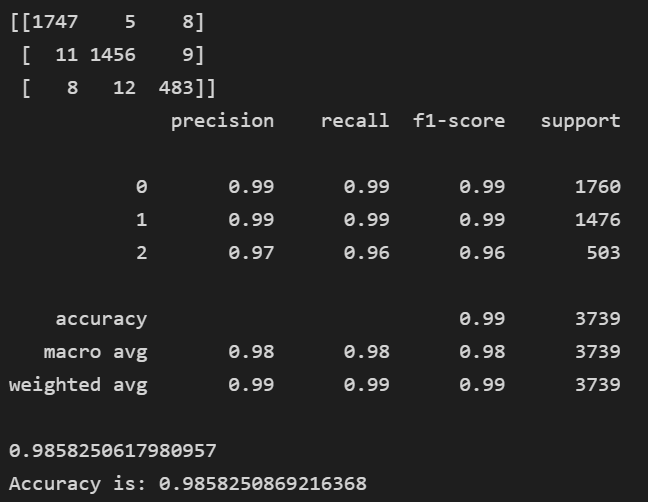
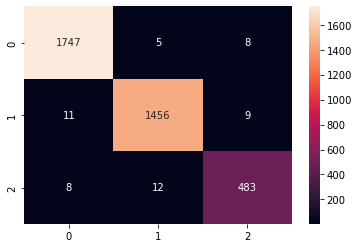


*Training and Validation Accuracy for BiLSTM Model*

Best Validation Accuracy : 0.98583

On Epoch : 65

Confusin Matrix :

# GRU

1. GRU Model

def model\_GRU(num\_words, embedding\_dim, maxlen):

    tf.random.set\_seed(123)

    gru\_dim = 32

    dense\_dim = 6

    modelGRU = tf.keras.Sequential([

        tf.keras.layers.Embedding(num\_words, embedding\_dim, input\_length=maxlen),

        tf.keras.layers.GRU(gru\_dim),

        tf.keras.layers.Dense(dense\_dim, activation='relu'),

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

    ])

    modelGRU.compile(loss='sparse\_categorical\_crossentropy',

                  optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

                  metrics=['accuracy'])

    return modelGRU

modelGRU = model\_GRU(NUM\_WORDS, EMBEDDING\_DIM, MAXLEN)

modelGRU.summary()

1. Model.fit

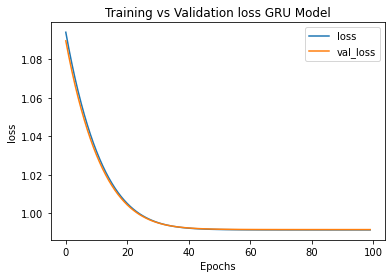
historyGRU = modelGRU.fit(train\_padded\_seq,

                        train\_label\_seq,

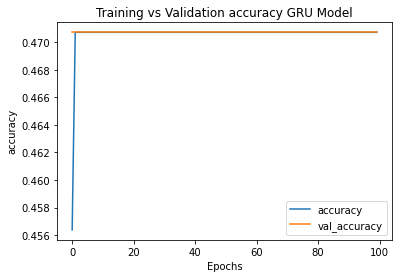
                        epochs=100,

                        validation\_data=(val\_padded\_seq, val\_label\_seq))

1. Result



*Training and Validation Loss for GRU Model*

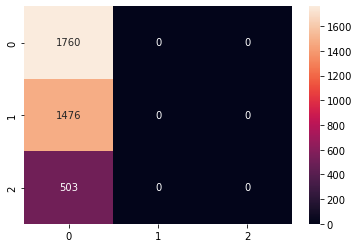
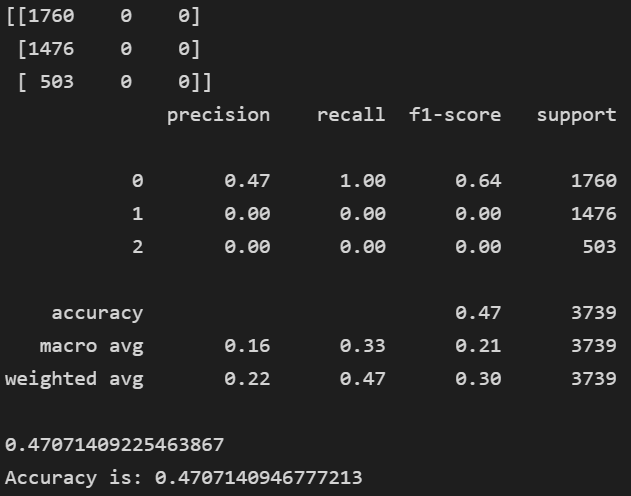


*Training and Validation Accuracy for GRU Model*

Best Validation Accuracy : 0.4707

On Epoch : 1

Confusin Matrix :



# BiGRU

1. BiGRU Model

def model\_BiGRU(num\_words, embedding\_dim, maxlen):

    tf.random.set\_seed(123)

    gru\_dim = 32

    dense\_dim = 6

    modelBiGRU = tf.keras.Sequential([

        tf.keras.layers.Embedding(num\_words, embedding\_dim, input\_length=maxlen),

        tf.keras.layers.Bidirectional(tf.keras.layers.GRU(gru\_dim)),

        tf.keras.layers.Dense(dense\_dim, activation='relu'),

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

    ])

    modelBiGRU.compile(loss='sparse\_categorical\_crossentropy',

                  optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

                  metrics=['accuracy'])

    return modelBiGRU

modelBiGRU = model\_BiGRU(NUM\_WORDS, EMBEDDING\_DIM, MAXLEN)

modelBiGRU.summary()

1. ModelCheckpoint & Model.fit

mc = ModelCheckpoint('classifier-putinG20-BiGRU\_04.h5', monitor='val\_accuracy', mode='max', verbose=1, save\_best\_only=True)

historyBiGRU = modelBiGRU.fit(train\_padded\_seq,

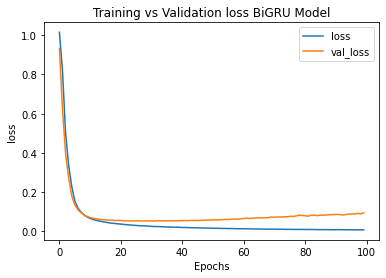
                            train\_label\_seq,

                            epochs=100,

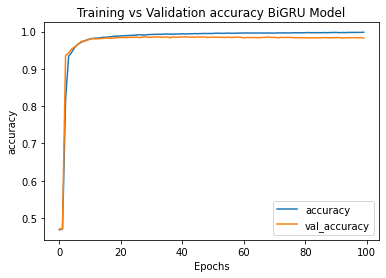
                            callbacks=[mc],

                            validation\_data=(val\_padded\_seq, val\_label\_seq))

1. Result



*Training and Validation Loss for BiGRU Model*

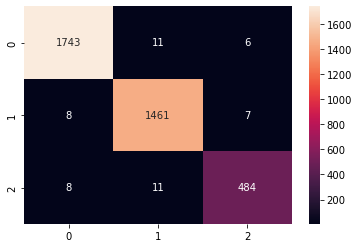
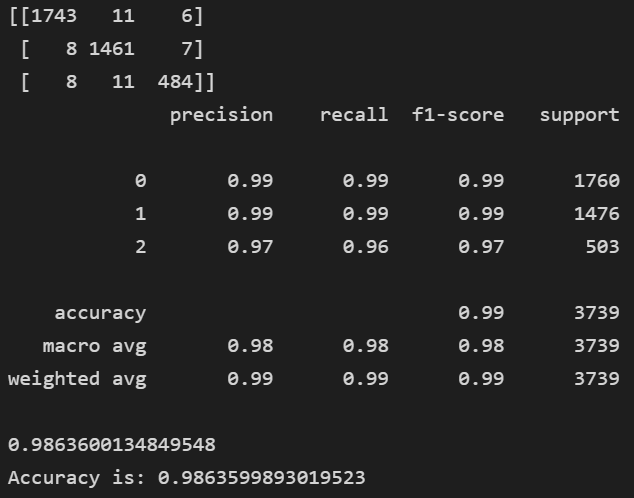


*Training and Validation Accuracy for BiGRU Model*

Best Validation Accuracy : 0.9864

On Epoch : 29

Confusin Matrix :



# CNN

1. CNN Model

def model\_conv(num\_words, embedding\_dim, maxlen):

    tf.random.set\_seed(123)

    filters = 64

    kernel\_size = 3

    dense\_dim = 6

    modelConv = tf.keras.Sequential([

        tf.keras.layers.Embedding(num\_words, embedding\_dim, input\_length=maxlen),

        tf.keras.layers.Conv1D(filters, kernel\_size, activation='relu'),

        tf.keras.layers.GlobalAveragePooling1D(),

        tf.keras.layers.Dense(dense\_dim, activation='relu'),

        tf.keras.layers.Dense(3, activation='sigmoid')

])

    modelConv.compile(loss='sparse\_categorical\_crossentropy',

                  optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

                  metrics=['accuracy'])

    return modelConv

modelConv = model\_conv(NUM\_WORDS, EMBEDDING\_DIM, MAXLEN)

modelConv.summary()

1. ModelCheckpoint & Model.fit

mc = ModelCheckpoint('classifier-putinG20-CNN\_01.h5', monitor='val\_accuracy', mode='max', verbose=1, save\_best\_only=True)

historyConv = modelConv.fit(train\_padded\_seq,

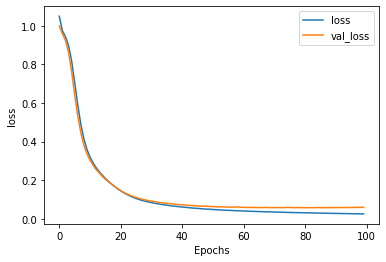
                            train\_label\_seq,

                            epochs=100,

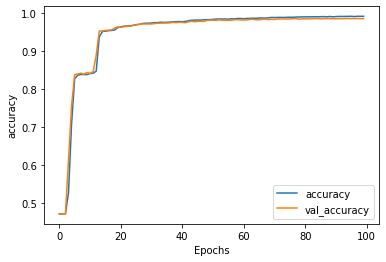
                            callbacks=[mc],

                            validation\_data=(val\_padded\_seq, val\_label\_seq))

1. Result



*Training and Validation Loss for CNN Model*

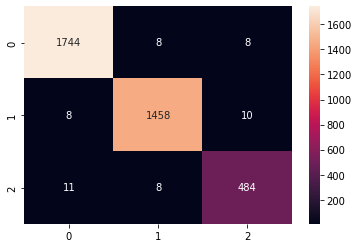
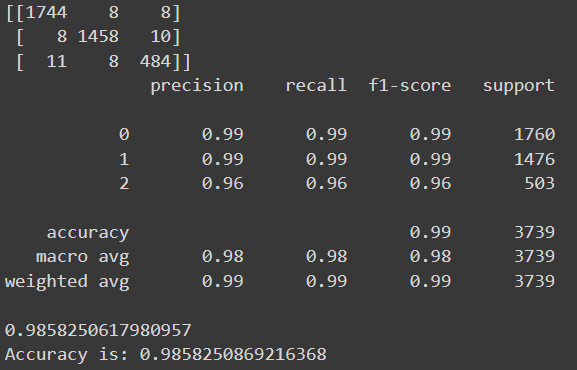


*Training and Validation Accuracy for CNN Model*

Best Validation Accuracy : 0.98583

On Epoch : 94

Confusin Matrix :



**SUMMARY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Model (Top Layer)*** | ***LSTM*** | ***BiLSTM*** | ***GRU*** | ***BiGRU*** | ***CNN*** |
| **Optimizer** | Adam | Adam | Adam | Adam | Adam |
| **Learning Rate** | 1e-4 | 1e-4 | 1e-4 | 1e-4 | 1e-4 |
| **Best Validation Accuracy** | 0.4707 | 0.98583 | 0.4707 | 0.9864 | 0.98583 |
| **On Epoch** | 1 | 65 | 1 | 29 | 94 |
| **Graph Train & Val Loss** |  |  |  |  |  |
| **Graph Train & Val Accuracy** |  |  |  |  |  |
| **Confusion Matrix** |  |  |  |  |  |