Laporan Tugas Akhir

PR 3

Helmi Alfarel

1806235896

Deskripsi Masalah

Diperoleh dataset yang berisi gambar wajah manusia(format .jpg), beserta dengan data umur, gender, dan etnis. Task yang ingin dilakukan adalah untuk memprediksi umur, gender, dan etnis seseorang menggunakan gambar wajahnya saja.

Dataset gambar wajah manusia tersebut memiliki spesifikasi sebagai berikut:
Jumlah gambar : 18964
Dimensi gambar : 48x48x1
Range nilai pada 1 channel gambar = 0,...,255

Terlihat bahwa data gambar yang diberikan merupakan gambar grayscale dengan ukuran 48x48. Sedangkan untuk masing-masing jenis label, memiliki spesifikasi sebagai berikut:

- Tipe data **umur** : numerik, diskret

- Tipe data **gender** : kategorikal
- Jumlah kelas **gender** : 2 (Male, Female)

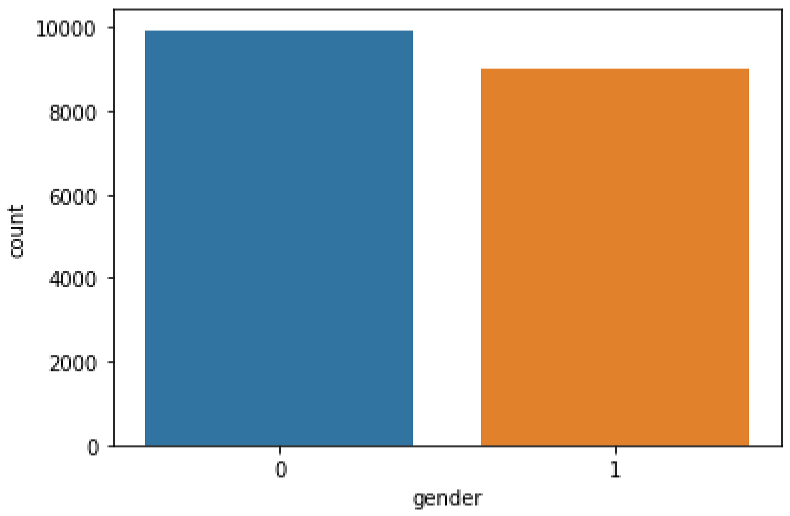
- Tipe data **etnis**: kategorikal
- Jumlah kelas **etnis** : 5 (White, Black, Asian, Indian, Others)

Metode yang paling umum untuk mengolah data gambar ke dalam Neural Networks adalah dengan Convolutional Neural Networks.

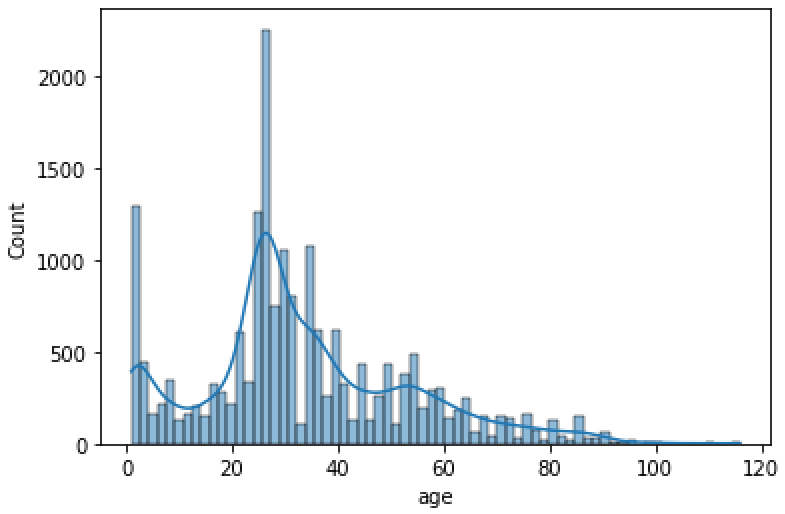
Exploratory data analysis (EDA)

Kita coba cek bagaimana distribusi datanya. Hal ini untuk mengetahui apakah data nya balance atau unbalance.

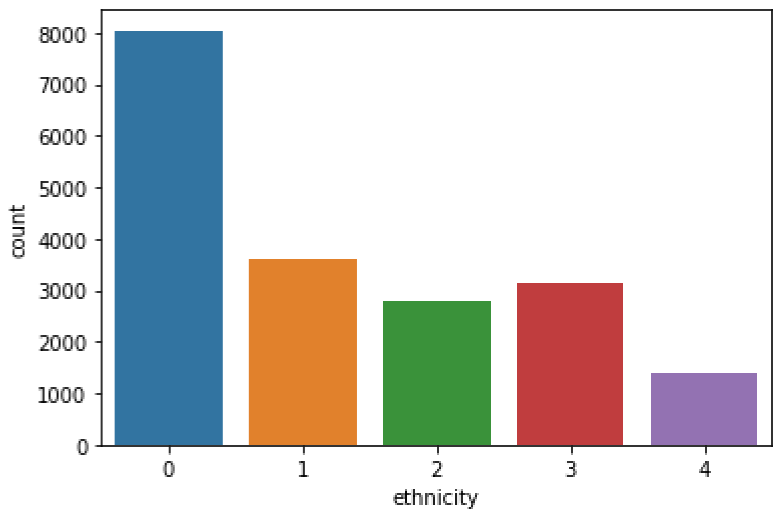
Berikut ada distribusi data gender. Bisa dilihat data gender cukup balance.



Berikut ada grafik distribusi umur. Terlihat bahwa datanya right-skewed dan terlihat terdapat 2 puncak.



Berikut adalah grafik distribusi etnis. Terlihat bahwa data etnis memiliki data White jauh lebih banyak daripada yang lain.



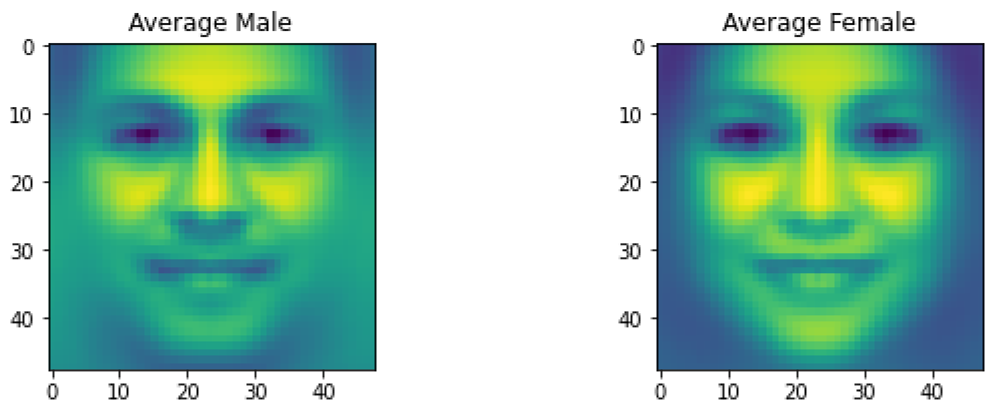
0 : White, 1: Black, 2: Asian, 3: Indian, 4:Others

Untuk data yang bertipe gambar, *EDA* yang bisa dilakukan cukup sedikit. Disini saya hanya menampilkan rata-rata dari data gambar wajah, sesuai dengan label dan kelas nya, kemudian dianalisa. Dan juga saya mengkombinasikan 2 label dan kelasnya.

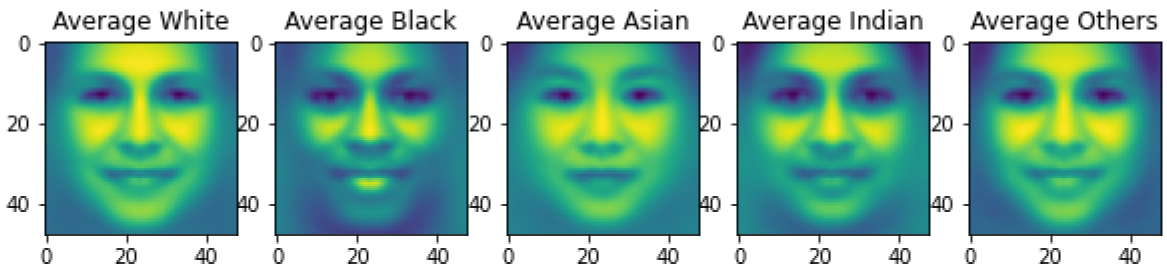
Kita melakukan EDA untuk mengecek apakah CNN mampu mendeteksi dan mengklasifikasi fitur-fitur wajah atau *Region-of-Interest*. Akan kita lihat dibawah nanti bahwa setiap kelas mempunyai fitur wajah yang unik, misalnya etnis Black biasanya mempunyai pixel lebih gelap dalam jumlah lebih banyak dibanding yang lain, atau Indian mempunyai bibir yang lebih gelap daripada yang lain. CNN diharapkan untuk dapat mendeteksi *Region-of-Interest*s seperti tersebut.

Berdasarkan gender, terlihat bahwa Female mempunyai warna hitam disekitar wajahnya, sedangkan Male tidak. Warna hitam itu adalah data rambut. CNN akan mampu mendeteksi apakah rambut di gambar wajah cukup banyak atau mengelilingi wajah atau tidak.

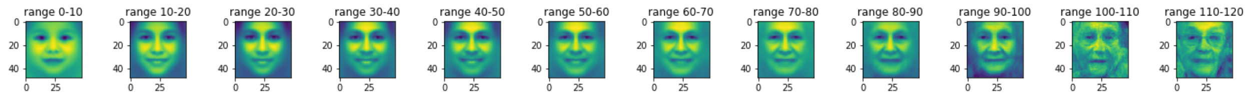
Note: EDA ini dilakukan sebelum preprocessing



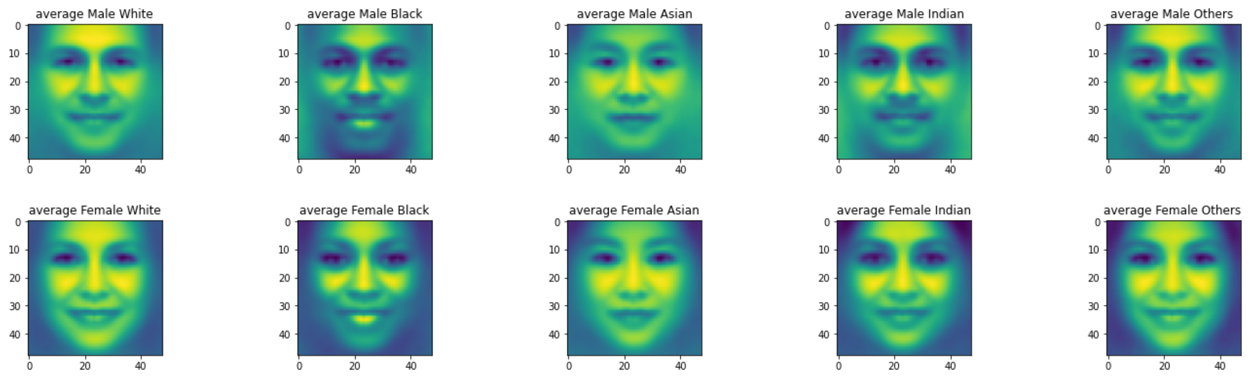
Berdasarkan etnis



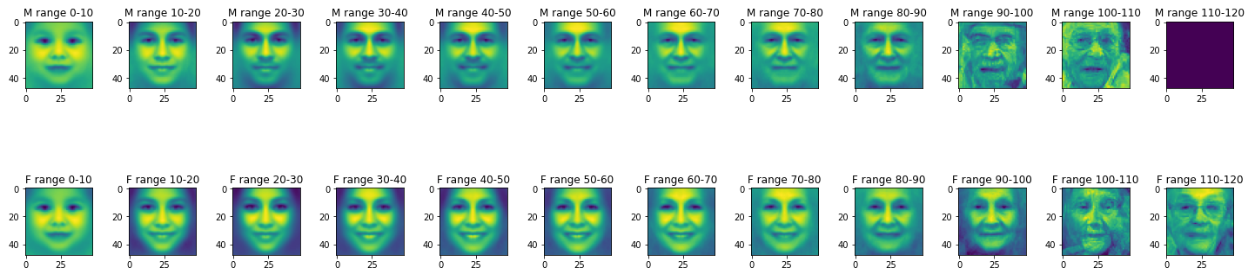
Berdasarkan umur



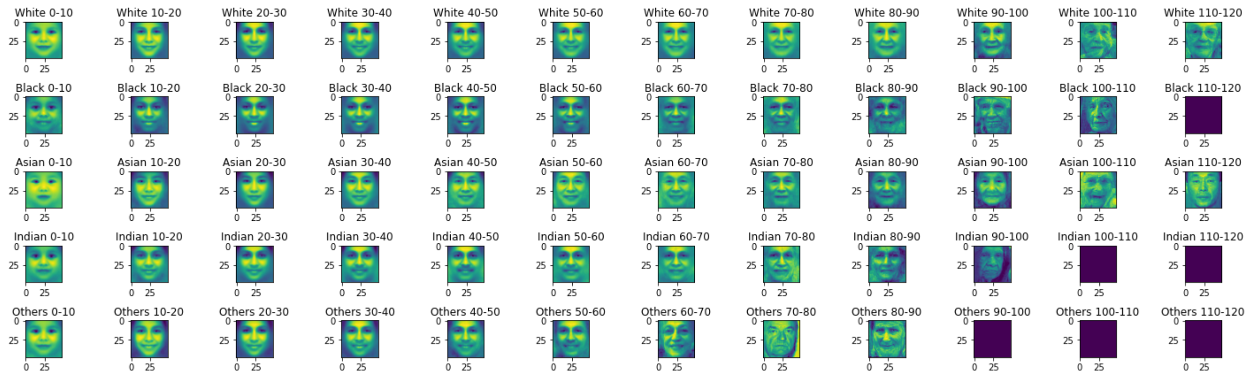
Berdasarkan gender & etnis



Berdasarkan gender & range usia



Berdasarkan etnis & range usia



Data Preprocessing

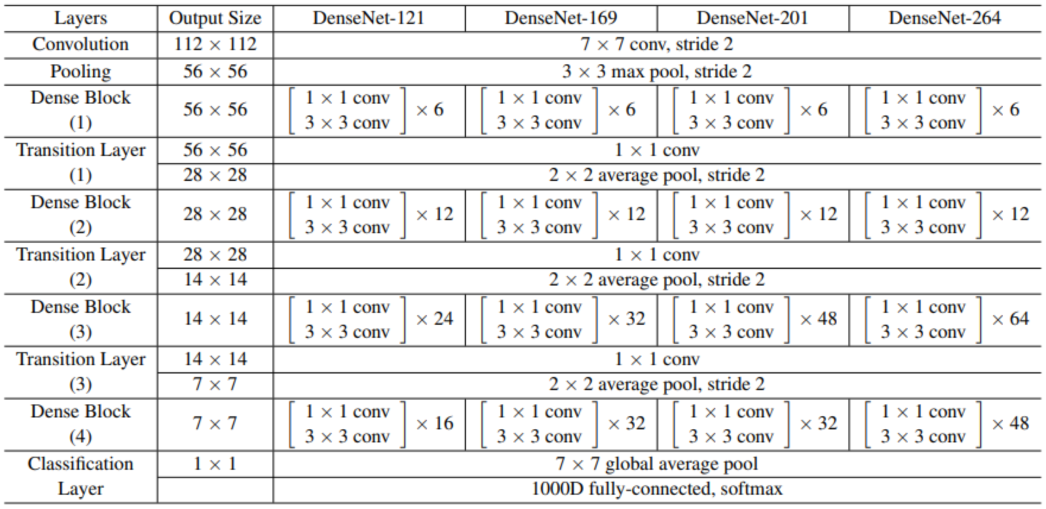
Data Preprocessing yang dilakukan pada data gambar adalah melakukan normalisasi pada nilai grayscale pada data gambar sehingga range nilai yang tadinya 0,..,255 menjadi 0,..,1. Hal ini dilakukan untuk mengurangi overfitting.

Preprocessing data umur dilakukan dengan cara mengubah tipe data dari numerik menjadi diskrit. Range setiap kelas pada data umur yang baru disesuaikan mengikuti *paper* oleh Levi and Hassner (2015)[2]

Model yang digunakan

Arsitektur yang saya gunakan pada tugas ini adalah CNN, lebih tepatnya menggunakan arsitektur DenseNet (Densely Connected Neural Networks). DenseNet terdiri dari beberapa Dense Block yang kemudian disambungkan dengan Fully Connected layer [1]. Dense Block mengolah input dengan cara melakukan konvolusi pada input tersebut namun hasil konvolusi tersebut di *concatenate* dengan input nya, yang kemudian akan dikeluarkan sebagai output. Akibatnya, dimensi panjang & lebar output Dense Block akan sama dengan input, namun dimensi *depth* output akan lebih besar. Dalam DenseNet bisa berisi beberapa Dense Block.

DenseNet memiliki beberapa arsitektur. Berikut adalah spesifikasi dari beberapa arsitektur Densenet:



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Model yang saya implementasikan pada tugas ini adalah modifikasi dari DenseNet121. Saya menambahkan *channel* baru pada output DenseBlock sehingga outputnya menjadi semakin dalam/tebal, namun saya mengurangi jumlah DenseBlock pada model.
Hasil yang diharapkan adalah berkurangnya space cost dan time cost.

Visualisasi DenseNet

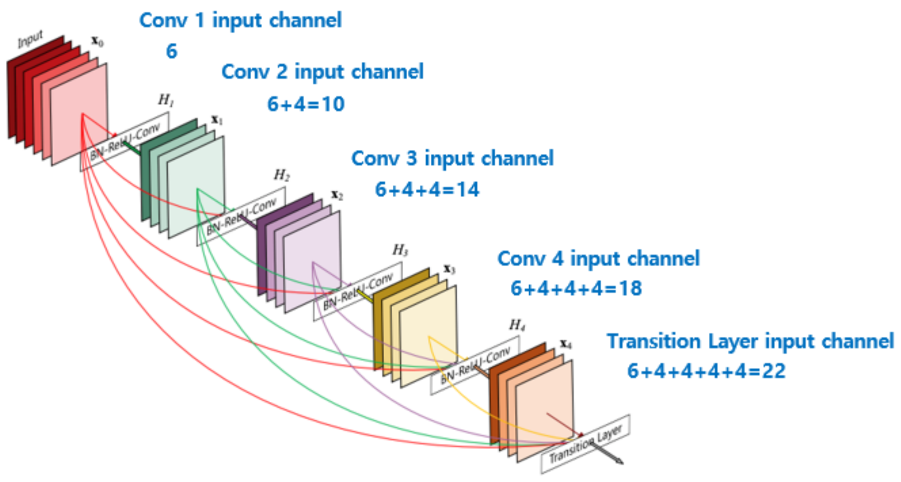
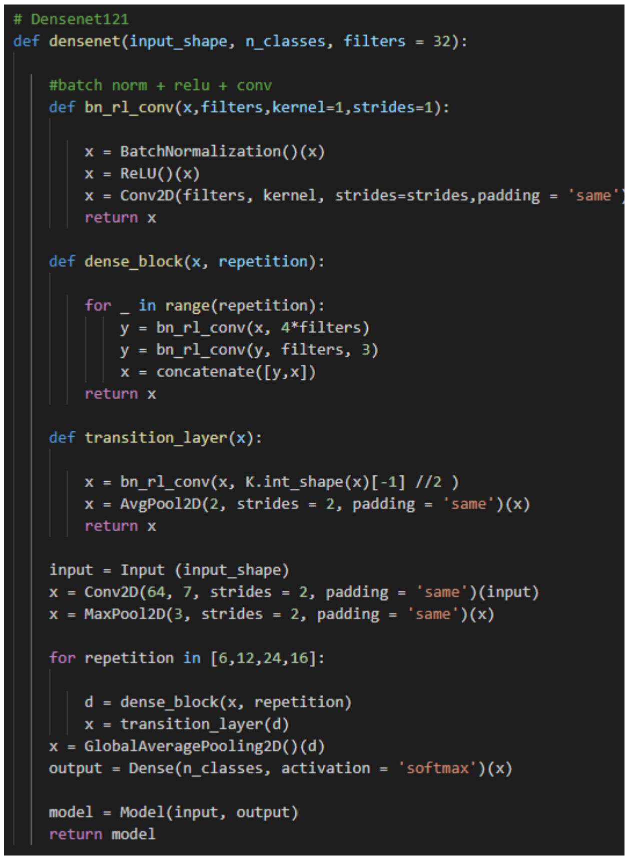


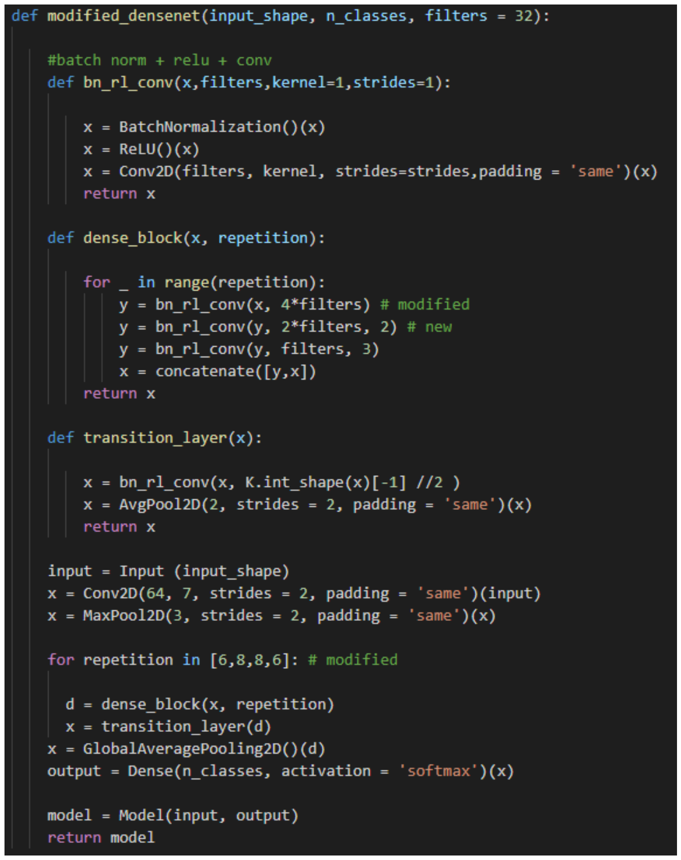
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Sumber: https://raw.githubusercontent.com/bruinxiong/densenet.mxnet/master/dense-block.png

Implementasi model Dense121 dan modified Densenet



```python
# Densenet121
def densenet(input_shape, n_classes, filters = 32):

    #batch norm + relu + conv
    def bn_rl_conv(x,filters,kernel=1,strides=1):

        x = BatchNormalization()(x)
        x = ReLU()(x)
        x = Conv2D(filters, kernel, strides=strides,padding = 'same'
        return x

    def dense_block(x, repetition):

        for _ in range(repetition):
            y = bn_rl_conv(x, 4*filters)
            y = bn_rl_conv(y, filters, 3)
            x = concatenate([y,x])
        return x

    def transition_layer(x):

        x = bn_rl_conv(x, K.int_shape(x)[-1] //2 )
        x = AvgPool2D(2, strides = 2, padding = 'same')(x)
        return x

    input = Input (input_shape)
    x = Conv2D(64, 7, strides = 2, padding = 'same')(input)
    x = MaxPool2D(3, strides = 2, padding = 'same')(x)

    for repetition in [6,12,24,16]:

        d = dense_block(x, repetition)
        x = transition_layer(d)
    x = GlobalAveragePooling2D()(d)
    output = Dense(n_classes, activation = 'softmax')(x)

    model = Model(input, output)
    return model
```



```python
def modified_densenet(input_shape, n_classes, filters = 32):

    #batch norm + relu + conv
    def bn_rl_conv(x,filters,kernel=1,strides=1):

        x = BatchNormalization()(x)
        x = ReLU()(x)
        x = Conv2D(filters, kernel, strides=strides,padding = 'same')(x)
        return x

    def dense_block(x, repetition):

        for _ in range(repetition):
            y = bn_rl_conv(x, 4*filters) # modified
            y = bn_rl_conv(y, 2*filters, 2) # new
            y = bn_rl_conv(y, filters, 3)
            x = concatenate([y,x])
        return x

    def transition_layer(x):

        x = bn_rl_conv(x, K.int_shape(x)[-1] //2 )
        x = AvgPool2D(2, strides = 2, padding = 'same')(x)
        return x

    input = Input (input_shape)
    x = Conv2D(64, 7, strides = 2, padding = 'same')(input)
    x = MaxPool2D(3, strides = 2, padding = 'same')(x)

    for repetition in [6,8,8,6]: # modified

      d = dense_block(x, repetition)
      x = transition_layer(d)
    x = GlobalAveragePooling2D()(d)
    output = Dense(n_classes, activation = 'softmax')(x)

    model = Model(input, output)
    return model
```

Hasil implementasi model

Spesifikasi :

Loss = sparse cross entropy

Optimizer = Adam

Persentase test data = 0.1

Epoch = 10

Gender

- DenseNet121

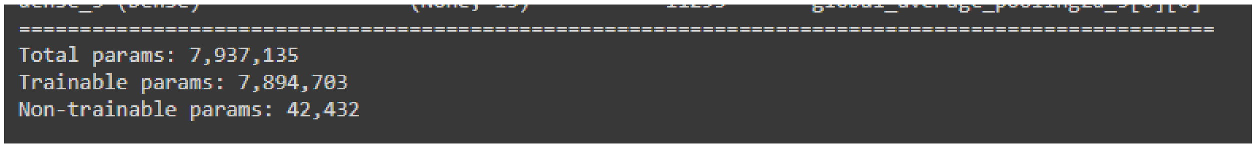
Jumlah Trainable params = 7.894.703

Training time / epoch = 33s

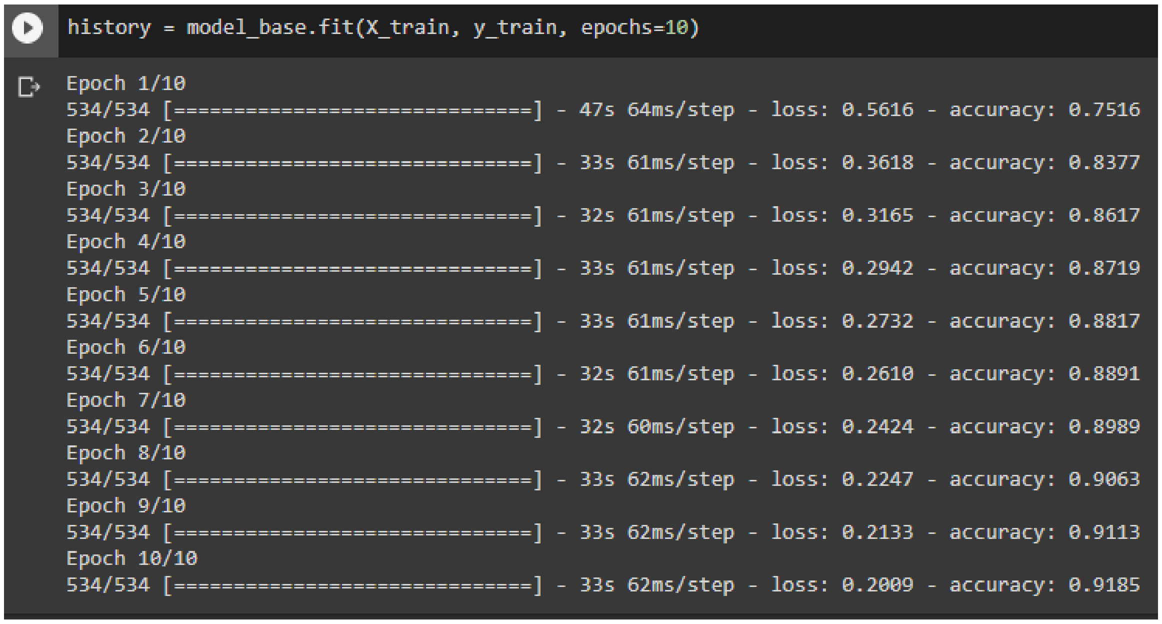
Training loss = 0.2009

Akurasi = 85%

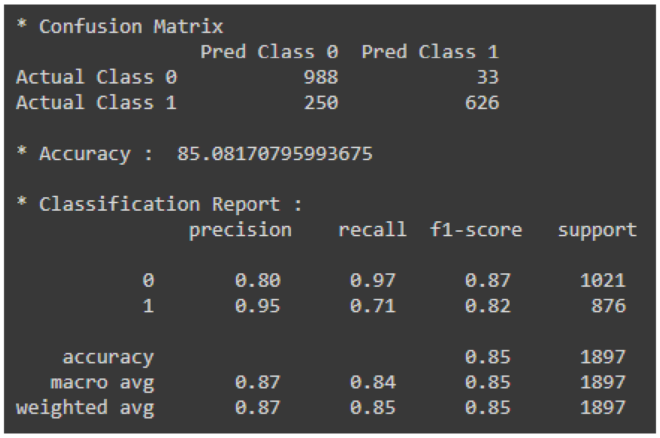
F1-score = 85%



```
==================================================================================================
Total params: 7,937,135
Trainable params: 7,894,703
Non-trainable params: 42,432
```



```
history = model_base.fit(X_train, y_train, epochs=10)

Epoch 1/10
534/534 [==============================] - 47s 64ms/step - loss: 0.5616 - accuracy: 0.7516
Epoch 2/10
534/534 [==============================] - 33s 61ms/step - loss: 0.3618 - accuracy: 0.8377
Epoch 3/10
534/534 [==============================] - 32s 61ms/step - loss: 0.3165 - accuracy: 0.8617
Epoch 4/10
534/534 [==============================] - 33s 61ms/step - loss: 0.2942 - accuracy: 0.8719
Epoch 5/10
534/534 [==============================] - 33s 61ms/step - loss: 0.2732 - accuracy: 0.8817
Epoch 6/10
534/534 [==============================] - 32s 61ms/step - loss: 0.2610 - accuracy: 0.8891
Epoch 7/10
534/534 [==============================] - 32s 60ms/step - loss: 0.2424 - accuracy: 0.8989
Epoch 8/10
534/534 [==============================] - 33s 62ms/step - loss: 0.2247 - accuracy: 0.9063
Epoch 9/10
534/534 [==============================] - 33s 62ms/step - loss: 0.2133 - accuracy: 0.9113
Epoch 10/10
534/534 [==============================] - 33s 62ms/step - loss: 0.2009 - accuracy: 0.9185
```



```
* Confusion Matrix
                Pred Class 0  Pred Class 1
Actual Class 0           988            33
Actual Class 1           250           626

* Accuracy :  85.08170795993675

* Classification Report :
               precision    recall  f1-score   support

           0       0.80      0.97      0.87      1021
           1       0.95      0.71      0.82       876

    accuracy                           0.85      1897
   macro avg       0.87      0.84      0.85      1897
weighted avg       0.87      0.85      0.85      1897
```

- ModifiedDenseNet

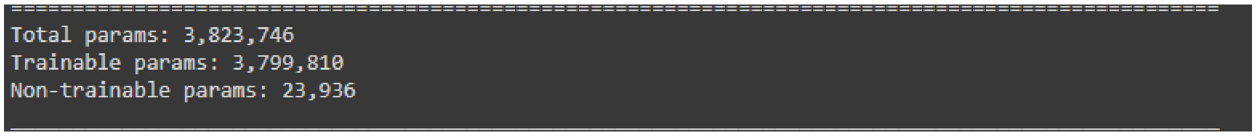
Jumlah Trainable params = 3.823.746

Avg training time per epoch = 14s

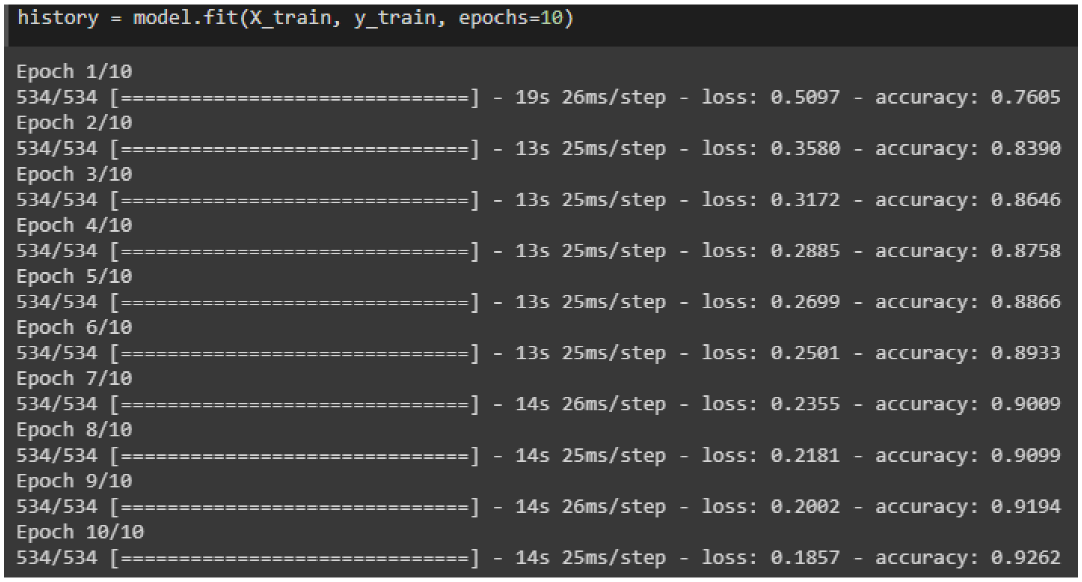
Training loss = 0.1857

Akurasi = 87%

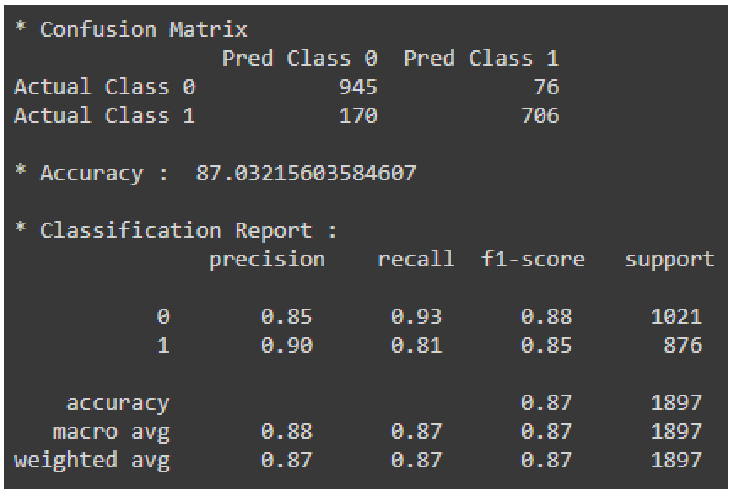
F1-score = 87%



```
==========================================================================================
Total params: 3,823,746
Trainable params: 3,799,810
Non-trainable params: 23,936
__________________________________________________________________________________________
```



```
history = model.fit(X_train, y_train, epochs=10)

Epoch 1/10
534/534 [==============================] - 19s 26ms/step - loss: 0.5097 - accuracy: 0.7605
Epoch 2/10
534/534 [==============================] - 13s 25ms/step - loss: 0.3580 - accuracy: 0.8390
Epoch 3/10
534/534 [==============================] - 13s 25ms/step - loss: 0.3172 - accuracy: 0.8646
Epoch 4/10
534/534 [==============================] - 13s 25ms/step - loss: 0.2885 - accuracy: 0.8758
Epoch 5/10
534/534 [==============================] - 13s 25ms/step - loss: 0.2699 - accuracy: 0.8866
Epoch 6/10
534/534 [==============================] - 13s 25ms/step - loss: 0.2501 - accuracy: 0.8933
Epoch 7/10
534/534 [==============================] - 14s 26ms/step - loss: 0.2355 - accuracy: 0.9009
Epoch 8/10
534/534 [==============================] - 14s 25ms/step - loss: 0.2181 - accuracy: 0.9099
Epoch 9/10
534/534 [==============================] - 14s 26ms/step - loss: 0.2002 - accuracy: 0.9194
Epoch 10/10
534/534 [==============================] - 14s 25ms/step - loss: 0.1857 - accuracy: 0.9262
```



```
* Confusion Matrix
                Pred Class 0  Pred Class 1
Actual Class 0           945            76
Actual Class 1           170           706

* Accuracy :  87.03215603584607

* Classification Report :
               precision    recall  f1-score   support

           0       0.85      0.93      0.88      1021
           1       0.90      0.81      0.85       876

    accuracy                           0.87      1897
   macro avg       0.88      0.87      0.87      1897
weighted avg       0.87      0.87      0.87      1897
```

Ethnicity

- DenseNet121

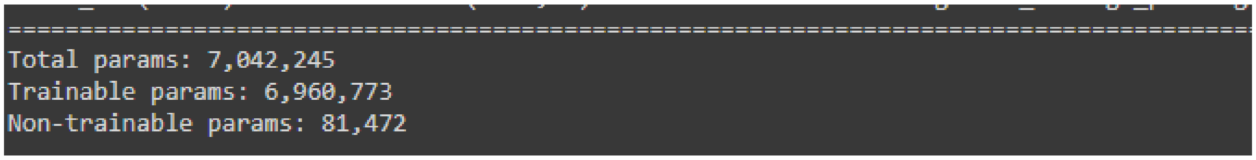
Jumlah Trainable params = 6.960.773

Avg training time per epoch = 35s

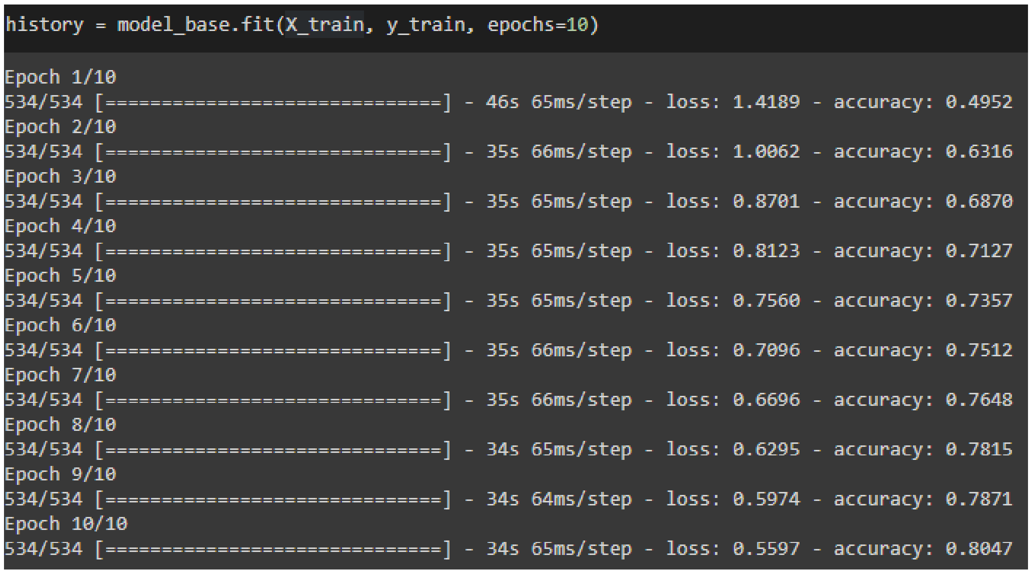
Training loss = 0.5597

Akurasi = 69%

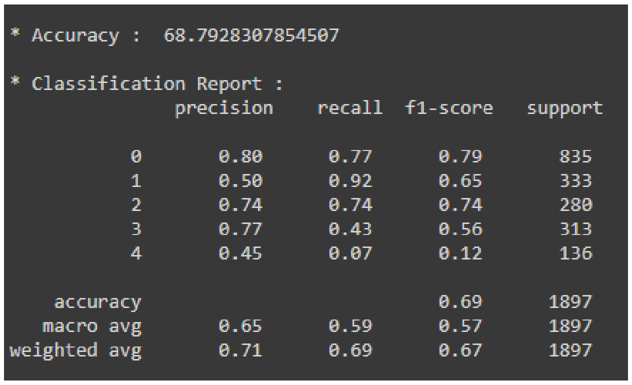
F1-score = 67%



```
================================================================================
Total params: 7,042,245
Trainable params: 6,960,773
Non-trainable params: 81,472
```



```
history = model_base.fit(X_train, y_train, epochs=10)

Epoch 1/10
534/534 [==============================] - 46s 65ms/step - loss: 1.4189 - accuracy: 0.4952
Epoch 2/10
534/534 [==============================] - 35s 66ms/step - loss: 1.0062 - accuracy: 0.6316
Epoch 3/10
534/534 [==============================] - 35s 65ms/step - loss: 0.8701 - accuracy: 0.6870
Epoch 4/10
534/534 [==============================] - 35s 65ms/step - loss: 0.8123 - accuracy: 0.7127
Epoch 5/10
534/534 [==============================] - 35s 65ms/step - loss: 0.7560 - accuracy: 0.7357
Epoch 6/10
534/534 [==============================] - 35s 66ms/step - loss: 0.7096 - accuracy: 0.7512
Epoch 7/10
534/534 [==============================] - 35s 66ms/step - loss: 0.6696 - accuracy: 0.7648
Epoch 8/10
534/534 [==============================] - 34s 65ms/step - loss: 0.6295 - accuracy: 0.7815
Epoch 9/10
534/534 [==============================] - 34s 64ms/step - loss: 0.5974 - accuracy: 0.7871
Epoch 10/10
534/534 [==============================] - 34s 65ms/step - loss: 0.5597 - accuracy: 0.8047
```


```
* Accuracy :  68.7928307854507

* Classification Report :
              precision    recall  f1-score   support

           0       0.80      0.77      0.79       835
           1       0.50      0.92      0.65       333
           2       0.74      0.74      0.74       280
           3       0.77      0.43      0.56       313
           4       0.45      0.07      0.12       136

    accuracy                           0.69      1897
   macro avg       0.65      0.59      0.57      1897
weighted avg       0.71      0.69      0.67      1897
```

- ModifiedDenseNet

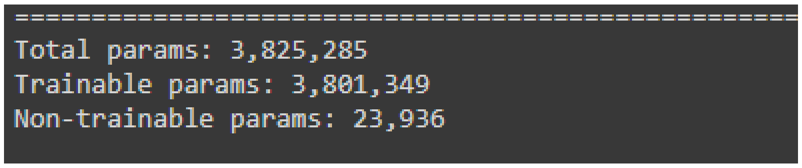
Jumlah Trainable params = 3.801.349

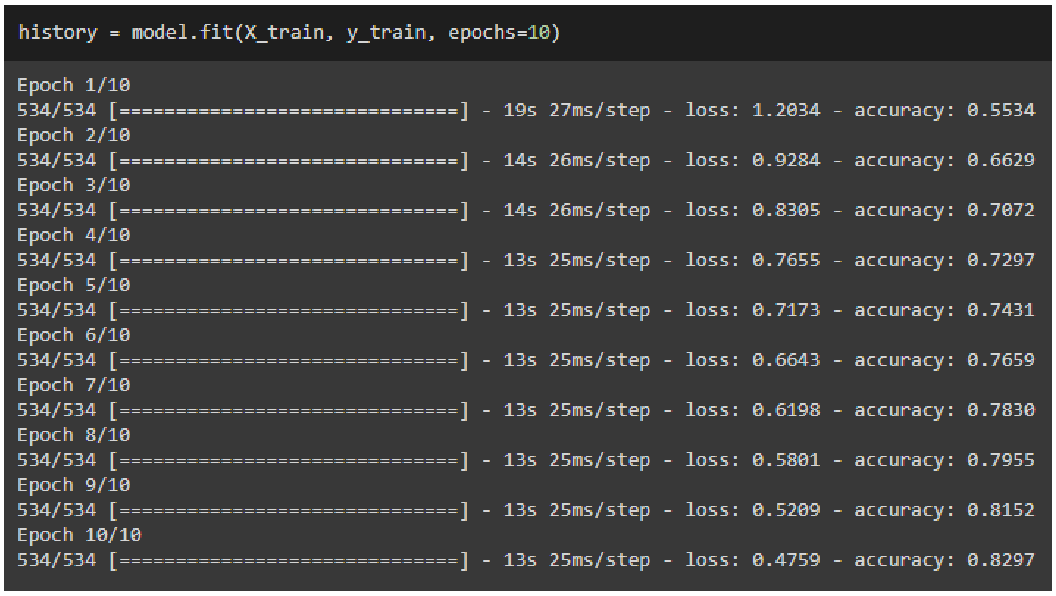
Avg training time per epoch = 13s

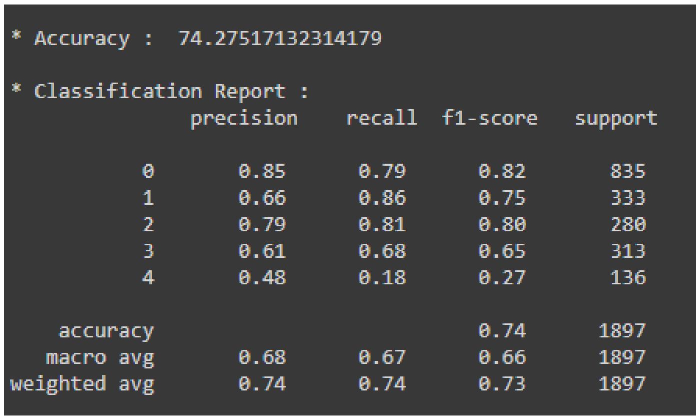
Training loss = 0.4759

Akurasi = 74%

F1-score = 73%


```
==================================================================
Total params: 3,825,285
Trainable params: 3,801,349
Non-trainable params: 23,936
```


```
history = model.fit(X_train, y_train, epochs=10)

Epoch 1/10
534/534 [==============================] - 19s 27ms/step - loss: 1.2034 - accuracy: 0.5534
Epoch 2/10
534/534 [==============================] - 14s 26ms/step - loss: 0.9284 - accuracy: 0.6629
Epoch 3/10
534/534 [==============================] - 14s 26ms/step - loss: 0.8305 - accuracy: 0.7072
Epoch 4/10
534/534 [==============================] - 13s 25ms/step - loss: 0.7655 - accuracy: 0.7297
Epoch 5/10
534/534 [==============================] - 13s 25ms/step - loss: 0.7173 - accuracy: 0.7431
Epoch 6/10
534/534 [==============================] - 13s 25ms/step - loss: 0.6643 - accuracy: 0.7659
Epoch 7/10
534/534 [==============================] - 13s 25ms/step - loss: 0.6198 - accuracy: 0.7830
Epoch 8/10
534/534 [==============================] - 13s 25ms/step - loss: 0.5801 - accuracy: 0.7955
Epoch 9/10
534/534 [==============================] - 13s 25ms/step - loss: 0.5209 - accuracy: 0.8152
Epoch 10/10
534/534 [==============================] - 13s 25ms/step - loss: 0.4759 - accuracy: 0.8297
```



```
* Accuracy :  74.27517132314179

* Classification Report :
               precision    recall  f1-score   support

           0       0.85      0.79      0.82       835
           1       0.66      0.86      0.75       333
           2       0.79      0.81      0.80       280
           3       0.61      0.68      0.65       313
           4       0.48      0.18      0.27       136

    accuracy                           0.74      1897
   macro avg       0.68      0.67      0.66      1897
weighted avg       0.74      0.74      0.73      1897
```

Age

- DenseNet121

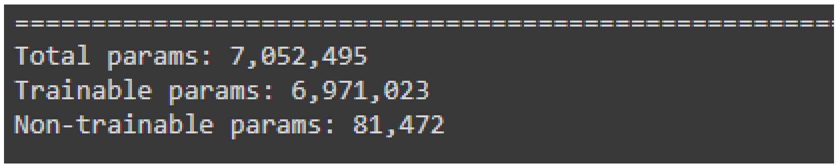
Jumlah Trainable params =6.971..023

Avg training time per epoch = 33s

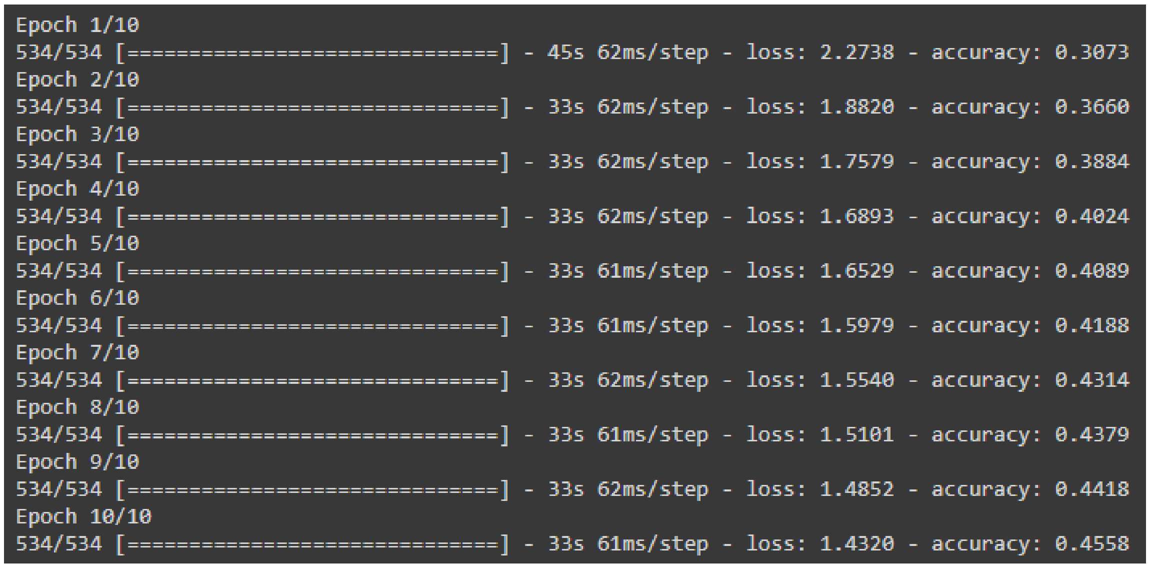
Training loss = 1.432

Akurasi = 40%

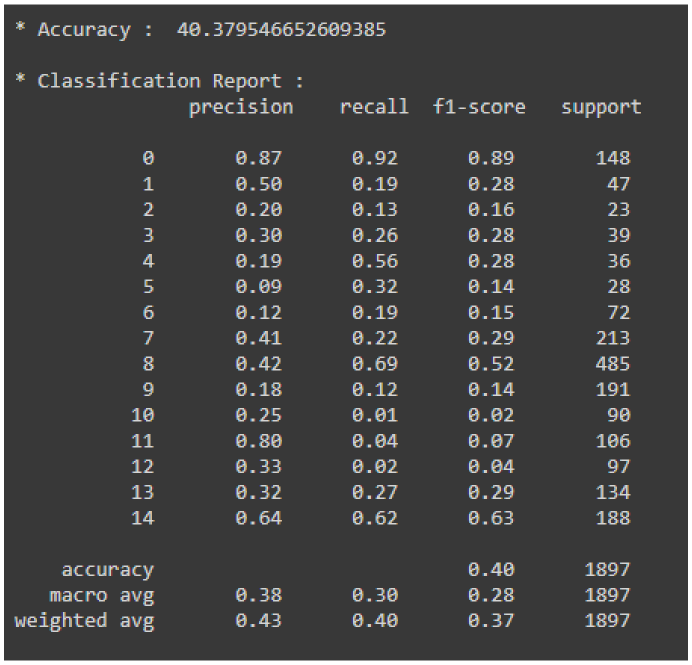
F1-score = 37%



```
=================================================================
Total params: 7,052,495
Trainable params: 6,971,023
Non-trainable params: 81,472
```



```
Epoch 1/10
534/534 [==============================] - 45s 62ms/step - loss: 2.2738 - accuracy: 0.3073
Epoch 2/10
534/534 [==============================] - 33s 62ms/step - loss: 1.8820 - accuracy: 0.3660
Epoch 3/10
534/534 [==============================] - 33s 62ms/step - loss: 1.7579 - accuracy: 0.3884
Epoch 4/10
534/534 [==============================] - 33s 62ms/step - loss: 1.6893 - accuracy: 0.4024
Epoch 5/10
534/534 [==============================] - 33s 61ms/step - loss: 1.6529 - accuracy: 0.4089
Epoch 6/10
534/534 [==============================] - 33s 61ms/step - loss: 1.5979 - accuracy: 0.4188
Epoch 7/10
534/534 [==============================] - 33s 62ms/step - loss: 1.5540 - accuracy: 0.4314
Epoch 8/10
534/534 [==============================] - 33s 61ms/step - loss: 1.5101 - accuracy: 0.4379
Epoch 9/10
534/534 [==============================] - 33s 62ms/step - loss: 1.4852 - accuracy: 0.4418
Epoch 10/10
534/534 [==============================] - 33s 61ms/step - loss: 1.4320 - accuracy: 0.4558
```


```
* Accuracy :  40.379546652609385

* Classification Report :
               precision    recall  f1-score   support

           0       0.87      0.92      0.89       148
           1       0.50      0.19      0.28        47
           2       0.20      0.13      0.16        23
           3       0.30      0.26      0.28        39
           4       0.19      0.56      0.28        36
           5       0.09      0.32      0.14        28
           6       0.12      0.19      0.15        72
           7       0.41      0.22      0.29       213
           8       0.42      0.69      0.52       485
           9       0.18      0.12      0.14       191
          10       0.25      0.01      0.02        90
          11       0.80      0.04      0.07       106
          12       0.33      0.02      0.04        97
          13       0.32      0.27      0.29       134
          14       0.64      0.62      0.63       188

    accuracy                           0.40      1897
   macro avg       0.38      0.30      0.28      1897
weighted avg       0.43      0.40      0.37      1897
```

- ModifiedDenseNet

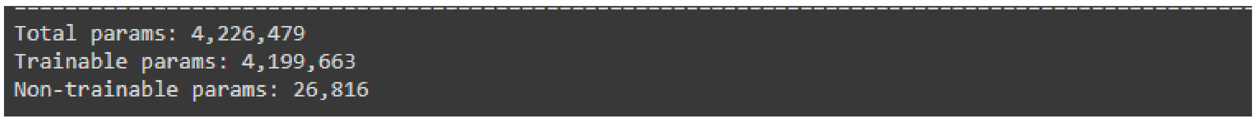
Jumlah Trainable params =4.199.663

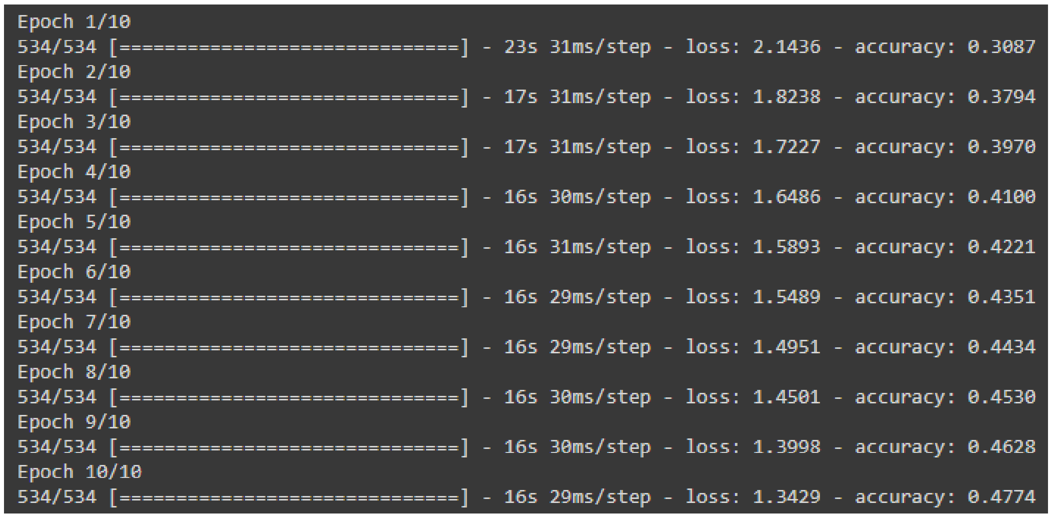
Avg training time per epoch = 16s

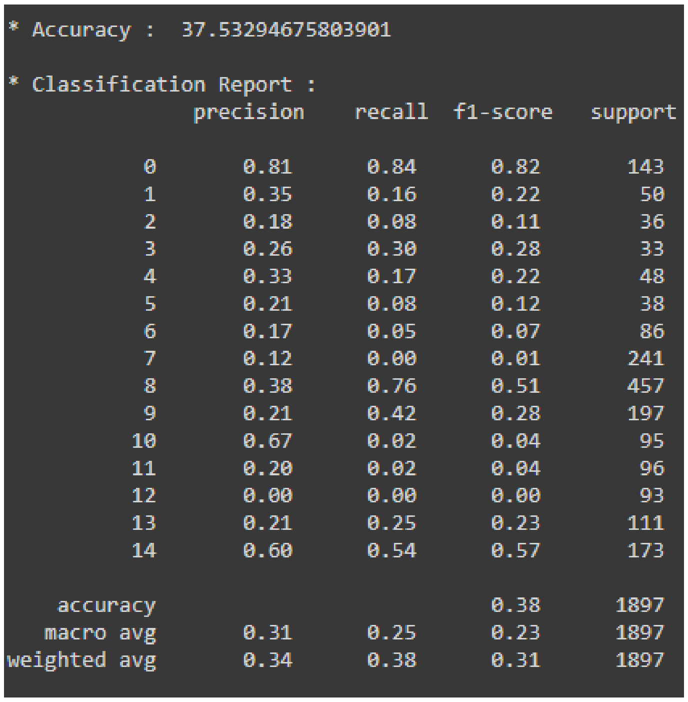
Training loss = 1.3429

Akurasi = 37%

F1-score = 31%


```
Total params: 4,226,479
Trainable params: 4,199,663
Non-trainable params: 26,816
```


```
Epoch 1/10
534/534 [==============================] - 23s 31ms/step - loss: 2.1436 - accuracy: 0.3087
Epoch 2/10
534/534 [==============================] - 17s 31ms/step - loss: 1.8238 - accuracy: 0.3794
Epoch 3/10
534/534 [==============================] - 17s 31ms/step - loss: 1.7227 - accuracy: 0.3970
Epoch 4/10
534/534 [==============================] - 16s 30ms/step - loss: 1.6486 - accuracy: 0.4100
Epoch 5/10
534/534 [==============================] - 16s 31ms/step - loss: 1.5893 - accuracy: 0.4221
Epoch 6/10
534/534 [==============================] - 16s 29ms/step - loss: 1.5489 - accuracy: 0.4351
Epoch 7/10
534/534 [==============================] - 16s 29ms/step - loss: 1.4951 - accuracy: 0.4434
Epoch 8/10
534/534 [==============================] - 16s 30ms/step - loss: 1.4501 - accuracy: 0.4530
Epoch 9/10
534/534 [==============================] - 16s 30ms/step - loss: 1.3998 - accuracy: 0.4628
Epoch 10/10
534/534 [==============================] - 16s 29ms/step - loss: 1.3429 - accuracy: 0.4774
```



```
* Accuracy :  37.53294675803901

* Classification Report :
              precision    recall  f1-score   support

           0       0.81      0.84      0.82       143
           1       0.35      0.16      0.22        50
           2       0.18      0.08      0.11        36
           3       0.26      0.30      0.28        33
           4       0.33      0.17      0.22        48
           5       0.21      0.08      0.12        38
           6       0.17      0.05      0.07        86
           7       0.12      0.00      0.01       241
           8       0.38      0.76      0.51       457
           9       0.21      0.42      0.28       197
          10       0.67      0.02      0.04        95
          11       0.20      0.02      0.04        96
          12       0.00      0.00      0.00        93
          13       0.21      0.25      0.23       111
          14       0.60      0.54      0.57       173

    accuracy                           0.38      1897
   macro avg       0.31      0.25      0.23      1897
weighted avg       0.34      0.38      0.31      1897
```

Kesimpulan

Ringkasan

	Gender		**Ethnicity**		**Age**	
	Dense121	Modified	Dense121	Modified	Dense121	Modified
# of Params	7.894.703	3.823.746	6.960.773	3.801.349	6.971.023	4.199.663
Avg training time / epoch (seconds)	33	14	35	13	33	16
Training Loss	0.2009	0.1857	0.5597	0.4759	1.432	1.3429
Akurasi	0.85	0.87	0.69	0.74	0.4	0.37
Macro F1-score	0.85	0.87	0.57	0.66	0.28	0.23
Weighted F1-Score	0.85	0.87	0.67	0.73	0.37	0.31

Analisis

Arsitektur Dense yang dimodifikasi dapat mengurangi jumlah parameter hingga setengah, dan dapat memotong waktu training hingga setengah dari arsitektur asli Dense121.

Dense yang dimodifikasi selalu menghasilkan training loss lebih kecil.

Pada case Gender dan Ethnicity, Dense yang dimodifikasi memiliki performa lebih baik daripada Dense121.

Pada case Age, walaupun training loss nya lebih kecil, performa Dense yang dimodifikasi lebih buruk daripada Dense121. Hal ini menunjukkan bahwa Dense yang dimodifikasi lebih *prone* terhadap overfitting.

Referensi

[1] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Published. https://doi.org/10.1109/cvpr.2017.243

[2] Levi, G., & Hassncer, T. (2015). Age and gender classification using convolutional neural networks. 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Published. https://doi.org/10.1109/cvprw.2015.7301352