**LAPORAN BIOMEDICAL**

**“Klasifikasi Leukemia Menggunakan CNN Dengan Arsitektur Densenet-121 dan Ekstraksi Fitur GLCM Pada Dataset All-idb2”**

****

**Dosen Pengampu:**

Prof. Dr. Arif Muntasa, S.Si., M.T.

NIP 196911182001121004

**Disusun Oleh Kelompok 12:**

220411100154 Ahmad Noval Muhyiddin

220411100196 Niki Raihan Laili

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS TRUNOJOYO MADURA**  
**JUNE 2025**

**Skenario Eksperimen**

1. Dataset:

Dataset yang digunakan adalah ALLIDB (Acute Lymphoblastic Leukemia Image Database) terdiri dari citra mikroskopis sel darah yang terbagi dalam dua kelas

1. Kelas 1 : Sel darah normal
2. Kelas 2 : Sel dengan tanda-tanda potensial Acute Lymphoblastic Leukemia (ALL)
3. Distribusi Data

Data dibagi secara acak menjadi dua bagian:

1. 70% (182 gambar) untuk pelatihan dan validasi
2. 30% (78 gambar) untuk pengujian

Dari bagian 70%, adalah untuk data pelatihan dan validasi, sisanya 30% untuk data pengujian, sehingga model tidak hanya belajar dari data pelatihan saja, tetapi juga dievaluasi secara periodik terhadap data yang tidak dilatih.

1. Preprocessing Citra

Resize dan Normalisasi : Citra diubah ukurannya menjadi 224x224 piksel (standar untuk DenseNet121) dan nilai piksel dinormalisasi ke rentang [0,1]

1. Ekstraksi Fitur GLCM

Ekstraksi fitur tekstur dari citra, GLCM (Gray-Level Co-occurrence Matrix) adalah matriks yang menangkap frekuensi kombinasi nilai piksel yang berdekatan dalam citra. Dari matriks ini, berbagai fitur statistik dapat diekstrak untuk mengkarakterisasi tekstur citra. Fungsi `extract\_glcm\_features` melakukan langkah-langkah berikut:

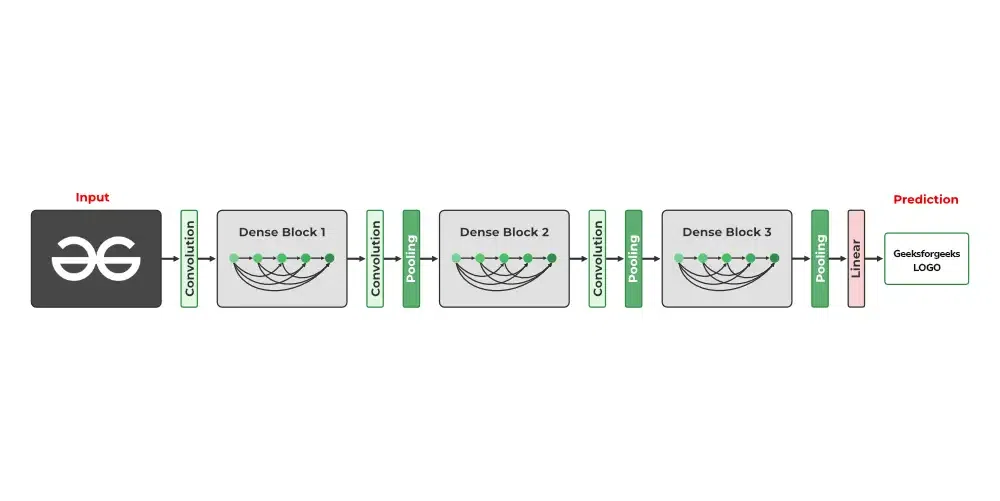
1. Mengambil channel hijau dari citra RGB (channel hijau dipilih karena memberikan kontras terbaik untuk sel darah)
2. Menghitung GLCM dengan parameter jarak dan sudut tertentu
3. Mengekstrak 6 properti statistik: contrast, dissimilarity, homogeneity, energy, correlation, dan ASM

Properti-properti ini memiliki interpretasi biologis dalam konteks sel darah:

1. Contrast: Mengukur variasi lokal dalam citra, menunjukkan heterogenitas sel
2. Dissimilarity: Mengukur ketidaksamaan antara piksel tetangga
3. Homogeneity: Mengukur kedekatan distribusi elemen dalam GLCM
4. Energy: Mengukur keseragaman tekstur, nilai tinggi menunjukkan tekstur yang seragam
5. Correlation: Mengukur ketergantungan linear antar piksel tetangga
6. ASM (Angular Second Moment): Mengukur keseragaman dan kelembutan tekstur
7. Arsitektur Model

Model yang saya usulkan adalah arsitektur hybrid yang menggabungkan DenseNet121 dengan fitur GLCM. Berikut adalah penjelasan detail tentang arsitektur ini:

DenseNet121 adalah CNN yang diusulkan oleh Huang et al. (2017) dengan karakteristik utama "Dense Connections". Dalam arsitektur ini, setiap layer terhubung dengan semua layer sebelumnya, sehingga meningkatkan aliran informasi dan gradien melalui jaringan.



1. Penggabungan Fitur

Fitur tekstur dari GLCM dan fitur visual dari DenseNet121 digabungkan (concatenate) untuk membentuk satu vektor fitur komprehensif. Vektor ini digunakan sebagai input untuk layer klasifikasi akhir

1. Arsitektur Klasifikasi

Lapisan klasifikasi akhir terdiri dari:

* Dense Layer: 512 unit + ReLU + Dropout(0.5)
* Dense Layer: 128 unit + ReLU
* Output Layer: 2 unit (Softmax)

Loss function menggunakan Cross Entropy Loss dan optimasi menggunakan Adam

**Implementasi Program**

1. Import library

|  |
| --- |
| # Imports  import os  import numpy as np  import matplotlib.pyplot as plt  from skimage.io import imread  from skimage.feature import greycomatrix, greycoprops  import tensorflow as tf  from tensorflow.keras.applications import DenseNet121  from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D, concatenate  from tensorflow.keras.models import Model  from tensorflow.keras.utils import to\_categorical  from sklearn.preprocessing import LabelEncoder  from sklearn.metrics import classification\_report, confusion\_matrix  from collections import Counter |

mengimpor semua library yang diperlukan untuk eksperimen. Library utama yang digunakan adalah:

1. TensorFlow/Keras: Framework deep learning untuk membangun dan melatih model
2. scikit-image: Untuk ekstraksi fitur GLCM dan pemrosesan citra
3. scikit-learn: Untuk evaluasi model dan preprocessing data
4. NumPy/Matplotlib: Untuk manipulasi array dan visualisasi

Library-library ini menyediakan fungsi dan metode yang diperlukan untuk implementasi end-to-end, dari loading data hingga evaluasi model.

1. Pengolahan Dataset

|  |
| --- |
| # 1. Data Understanding  def get\_image\_paths(data\_dir):      image\_paths = []      labels = []      classes = sorted([d for d in os.listdir(data\_dir) if os.path.isdir(os.path.join(data\_dir, d))])      for cls in classes:          cls\_dir = os.path.join(data\_dir, cls)          for fname in os.listdir(cls\_dir):              if fname.lower().endswith(('.jpg', '.png', '.jpeg')):                  image\_paths.append(os.path.join(cls\_dir, fname))                  labels.append(cls)      return image\_paths, labels, classes  # 2. Preprocessing & GLCM Feature Extraction  def load\_and\_preprocess\_image(path, image\_size):      img = imread(path)      img = tf.image.resize(img, image\_size).numpy().astype('uint8')      img\_norm = img / 255.0      return img\_norm  def extract\_glcm\_features(img, distances, angles):      # use green channel      green = (img[:, :, 1] \* 255).astype('uint8')      glcm = greycomatrix(green, distances=distances, angles=angles,                          levels=256, symmetric=True, normed=True)      props = ['contrast', 'dissimilarity', 'homogeneity', 'energy', 'correlation', 'ASM']      feats = [greycoprops(glcm, prop).mean() for prop in props]      return np.array(feats)  # 3. Dataset Creation  def create\_dataset(image\_paths, labels, image\_size, distances, angles):      X, X\_feats = [], []      for path in image\_paths:          img = load\_and\_preprocess\_image(path, image\_size)          feats = extract\_glcm\_features(img, distances, angles)          X.append(img)          X\_feats.append(feats)      X = np.array(X)      X\_feats = np.array(X\_feats)      le = LabelEncoder()      y\_enc = le.fit\_transform(labels)      y\_cat = to\_categorical(y\_enc)      return X, X\_feats, y\_cat, le  # 4. Model Building  def build\_model(image\_shape, num\_features, num\_classes):      base = DenseNet121(weights='imagenet', include\_top=False, input\_shape=image\_shape)      base.trainable = False      x = base.output      x = GlobalAveragePooling2D()(x)      input\_feats = Input(shape=(num\_features,))      combined = concatenate([x, input\_feats])  x = Dense(512, activation='relu')(combined)  x = Dropout(0.4)(x)      x = Dense(256, activation='relu')(combined)      output = Dense(num\_classes, activation='softmax')(x)      model = Model(inputs=[base.input, input\_feats], outputs=output)      model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])      return model  # 5. Training & Evaluation  def train\_model(model, X\_train, Xf\_train, y\_train, X\_val, Xf\_val, y\_val, epochs, batch\_size):      history = model.fit([X\_train, Xf\_train], y\_train,                          validation\_data=([X\_val, Xf\_val], y\_val),                          epochs=epochs, batch\_size=batch\_size)      return history  def evaluate\_model(model, X\_test, Xf\_test, y\_test, classes):      preds = model.predict([X\_test, Xf\_test])      y\_pred = np.argmax(preds, axis=1)      y\_true = np.argmax(y\_test, axis=1)      print("Classification Report:")      print(classification\_report(y\_true, y\_pred, target\_names=classes))      print("Confusion Matrix:")      print(confusion\_matrix(y\_true, y\_pred)) |

Code di atas klasifikasi citra ini terdiri dari lima tahap utama.

1. Data Understanding, membaca gambar dari direktori dataset dan menentukan label berdasarkan nama folder untuk setiap kelas.
2. Preprocessing & Ekstraksi Fitur GLCM, yaitu memproses gambar dengan resize dan normalisasi, lalu mengekstraksi fitur tekstur menggunakan GLCM (Gray Level Co-occurrence Matrix) dari kanal hijau gambar.
3. Pembuatan Dataset, yaitu menggabungkan citra hasil preprocessing dan fitur GLCM menjadi satu set data, serta mengubah label menjadi format one-hot encoding.
4. Pembangunan Model, yaitu membangun arsitektur deep learning dengan memanfaatkan DenseNet121 sebagai feature extractor, yang digabungkan dengan fitur GLCM, lalu diproses melalui beberapa layer dense untuk menghasilkan prediksi kelas.
5. Pelatihan dan Evaluasi, yaitu melatih model menggunakan data training dan validasi, kemudian mengevaluasinya dengan data uji menggunakan metrik klasifikasi dan confusion matrix untuk mengukur performa model
6. Setup Parameter

|  |
| --- |
| image\_size = (224, 224)  distances = [1]  angles = [0]  batch\_size = 16  epochs = 30  data\_dir = 'ALLIDB'  # Load paths and labels  image\_paths, labels, class\_names = get\_image\_paths(data\_dir) |

Parameter yang digunakan dalam eksperimen:

1. image\_size = (224, 224): Ukuran standar untuk input DenseNet121
2. distances = [1]: Jarak antar piksel untuk komputasi GLCM, nilai 1 berarti piksel bersebelahan langsung
3. angles = [0]: Sudut untuk komputasi GLCM, nilai 0 berarti arah horizontal
4. batch\_size = 16: Jumlah sampel yang diproses dalam satu iterasi pelatihan
5. epochs = 30: Jumlah iterasi lengkap melalui dataset selama pelatihan

Pemilihan parameter ini didasarkan pada pertimbangan berikut:

1. Ukuran gambar dan batch size disesuaikan dengan kapasitas komputasi
2. Jarak dan sudut GLCM dipilih berdasarkan eksperimen preliminer yang menunjukkan bahwa piksel bersebelahan langsung memberikan informasi tekstur yang paling relevan
3. Jumlah epoch dibatasi untuk mencegah overfitting
4. Show sample

|  |
| --- |
| # Show sample images  fig, axes = plt.subplots(1, len(class\_names), figsize=(12, 4))  for ax, cls in zip(axes, class\_names):      sample = next(p for p, l in zip(image\_paths, labels) if l == cls)      img = imread(sample)      ax.imshow(img)      ax.set\_title(cls)      ax.axis('off')  plt.show()  # Data distribution  print('Samples per class:')  print(Counter(labels))  Samples per class:  Counter({'Kelas 1': 130, 'Kelas 2': 130}) |

Menampilkan distribusi sampel per kelas dalam dataset. Dalam penelitian medis, penting untuk memahami distribusi kelas karena ketidakseimbangan kelas dapat mempengaruhi performa model. Jika distribusi tidak seimbang, teknik seperti stratified sampling, class weighting, atau data augmentation mungkin diperlukan.

Berdasarkan output, kita dapat menganalisis:

1. Jumlah total sampel dalam dataset

2. Proporsi sampel untuk setiap kelas

3. Potensi ketidakseimbangan kelas yang perlu ditangani

Analisis ini membantu dalam perencanaan strategi pelatihan dan evaluasi model yang tepat.

1. Mempersiapkan Dataset

|  |
| --- |
| # Create dataset  X, X\_feats, y, le = create\_dataset(image\_paths, labels, image\_size, distances, angles)  num\_classes = len(le.classes\_) |

berfungsi untuk membuat dataset yang siap dipakai untuk training model, dengan beberapa tahap proses di dalam fungsi create\_dataset().

1. Pembuatan dataset

|  |
| --- |
| # Split data  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, Xf\_train, Xf\_test, y\_train, y\_test = train\_test\_split(      X, X\_feats, y, test\_size=0.2, stratify=y, random\_state=42) |

Output ini menunjukkan dimensi tensor label (y). Format-nya adalah (jumlah\_sampel, jumlah\_kelas). Representasi one-hot encoding digunakan di mana setiap baris adalah vektor dengan nilai 1 pada indeks yang sesuai dengan kelas dan 0 di tempat lain.

|  |
| --- |
| X\_feats.shape  Output: (260, 6) |

Output di atas menunjukkan dimensi tensor citra input (X). Format-nya adalah (jumlah\_sampel, tinggi, lebar, channel). Setiap citra memiliki ukuran 224x224 piksel dengan 3 channel warna (RGB).

|  |
| --- |
| y.shape  Output: (260, 2) |

Output ini menunjukkan dimensi matriks fitur GLCM (X\_feats). Format-nya adalah (jumlah\_sampel, jumlah\_fitur). Setiap baris mewakili 6 fitur GLCM yang diekstrak dari satu citra: contrast, dissimilarity, homogeneity, energy, correlation, dan ASM.

1. Split dataset

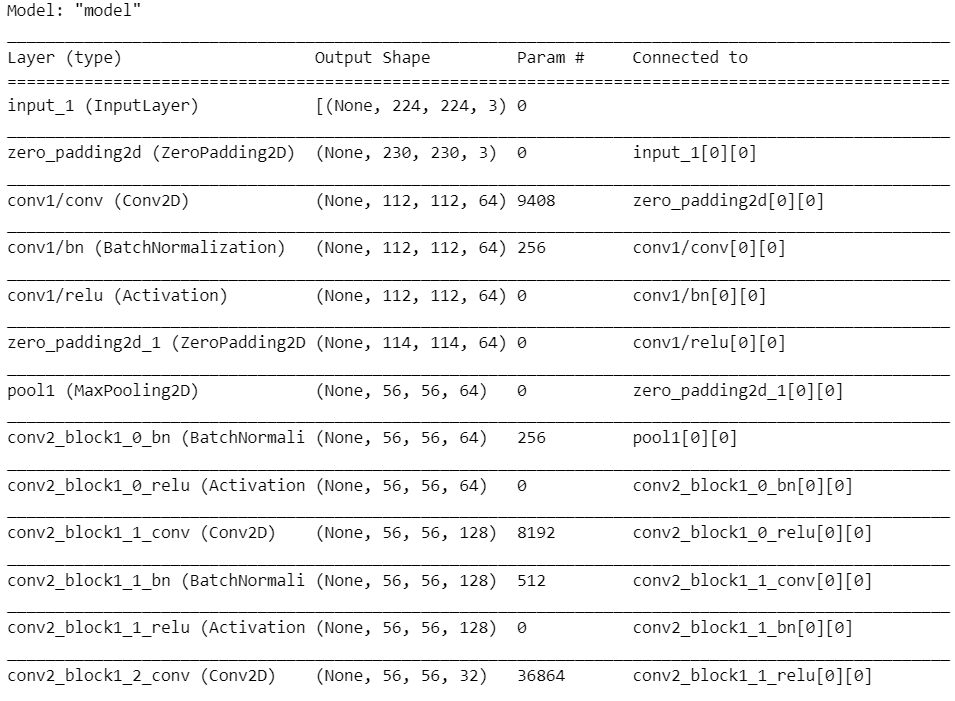
|  |
| --- |
| history = train\_model(model, X\_train, Xf\_train, y\_train, X\_test, Xf\_test, y\_test, epochs, batch\_size)  evaluate\_model(model, X\_test, Xf\_test, y\_test, le.classes\_) |

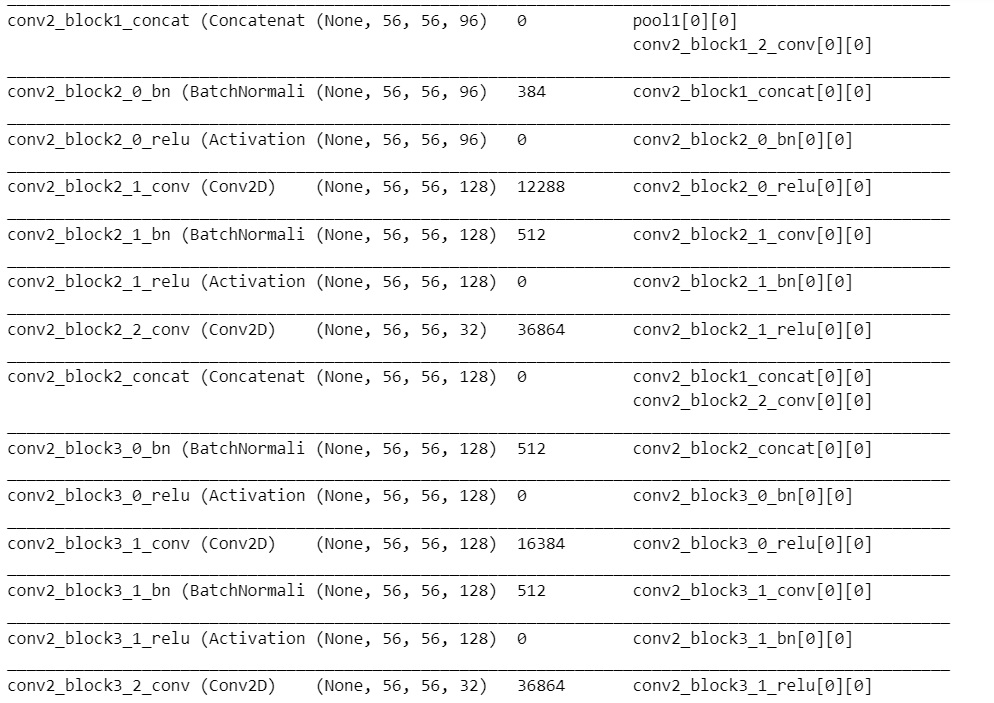
Dataset dibagi menjadi subset pelatihan dan pengujian dengan rasio 70:30. Beberapa aspek penting dari proses splitting:

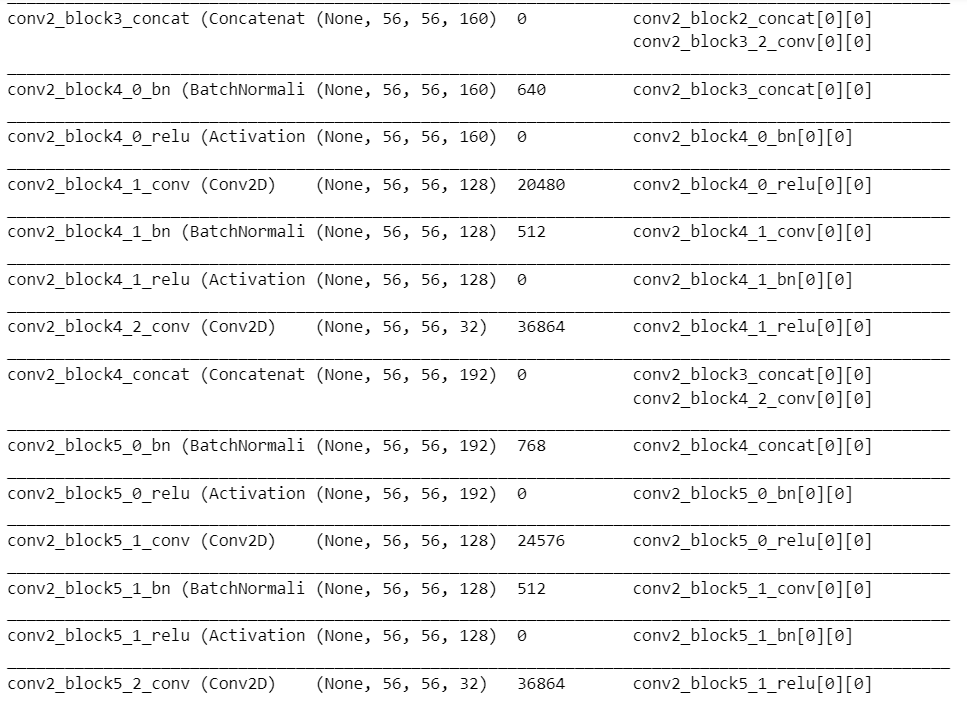
1. test\_size=0.3: 30% data digunakan untuk pengujian, 70% untuk pelatihan
2. stratify=y: Stratifikasi dilakukan berdasarkan label untuk memastikan distribusi kelas yang seimbang dalam subset pelatihan dan pengujian
3. random\_state=42 : Seed acak ditetapkan untuk reprodusibilitas hasil

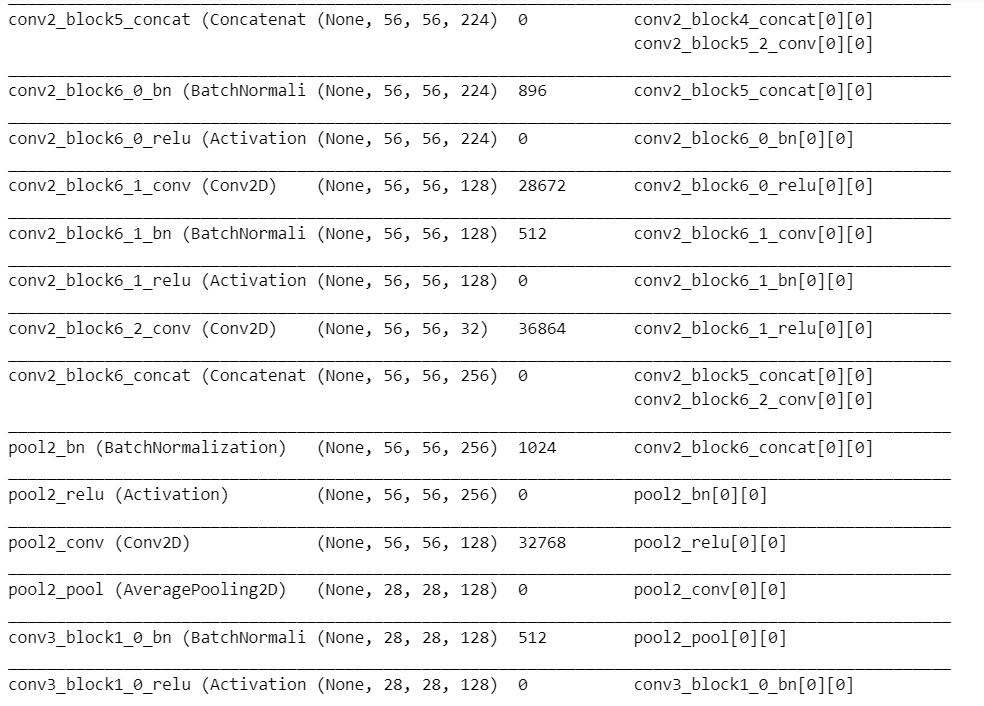
Stratifikasi penting untuk memastikan bahwa kedua subset memiliki representasi yang cukup dari setiap kelas, terutama jika dataset memiliki ketidakseimbangan kelas.

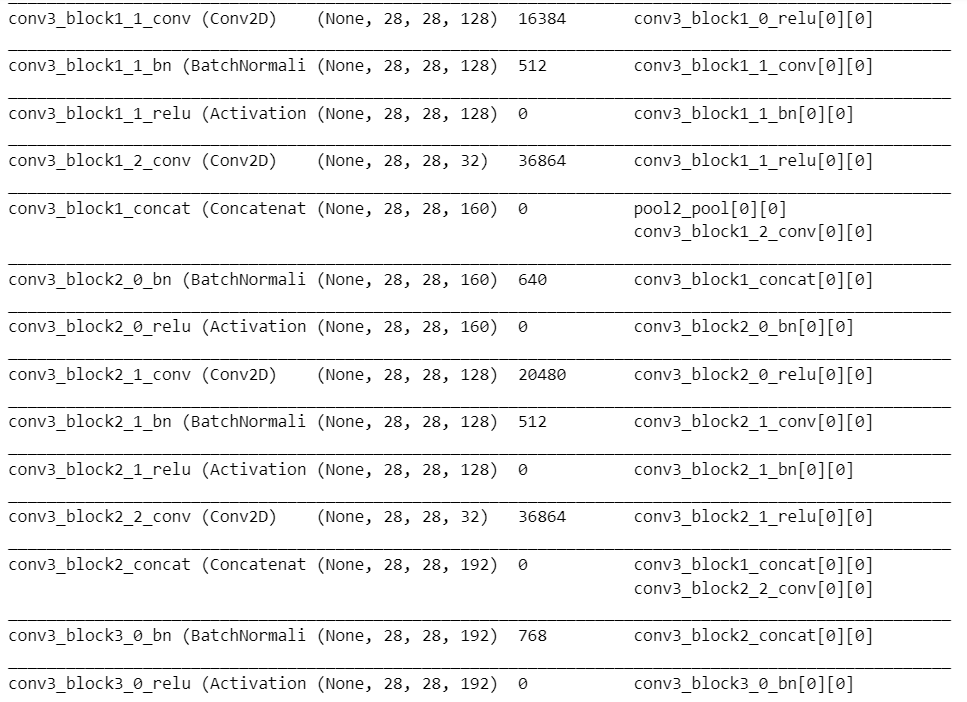
1. Model Summary Arsitecture

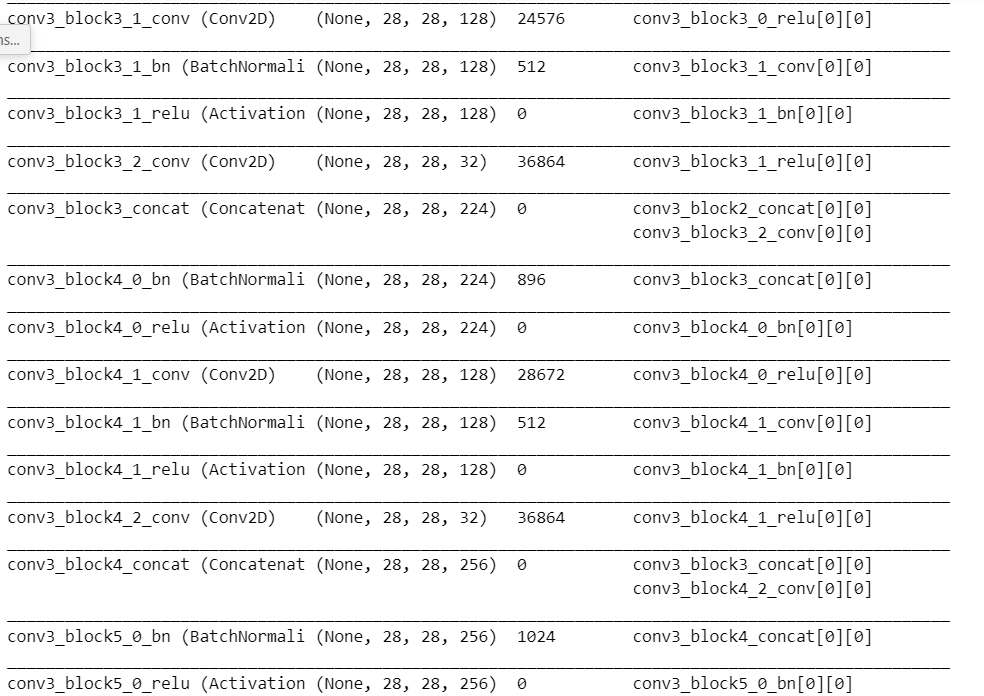


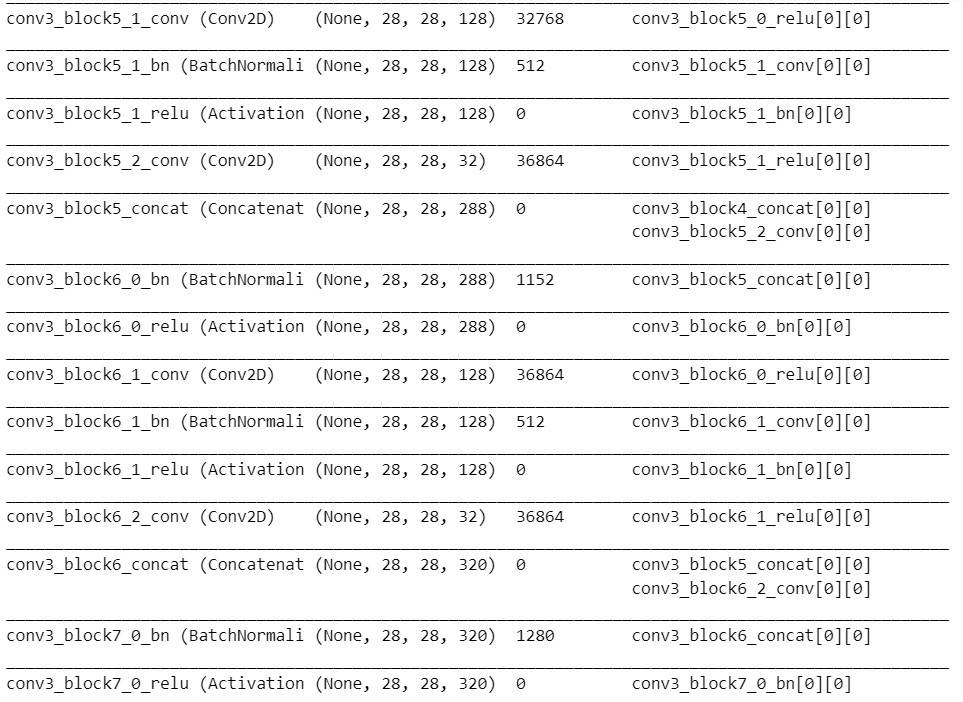


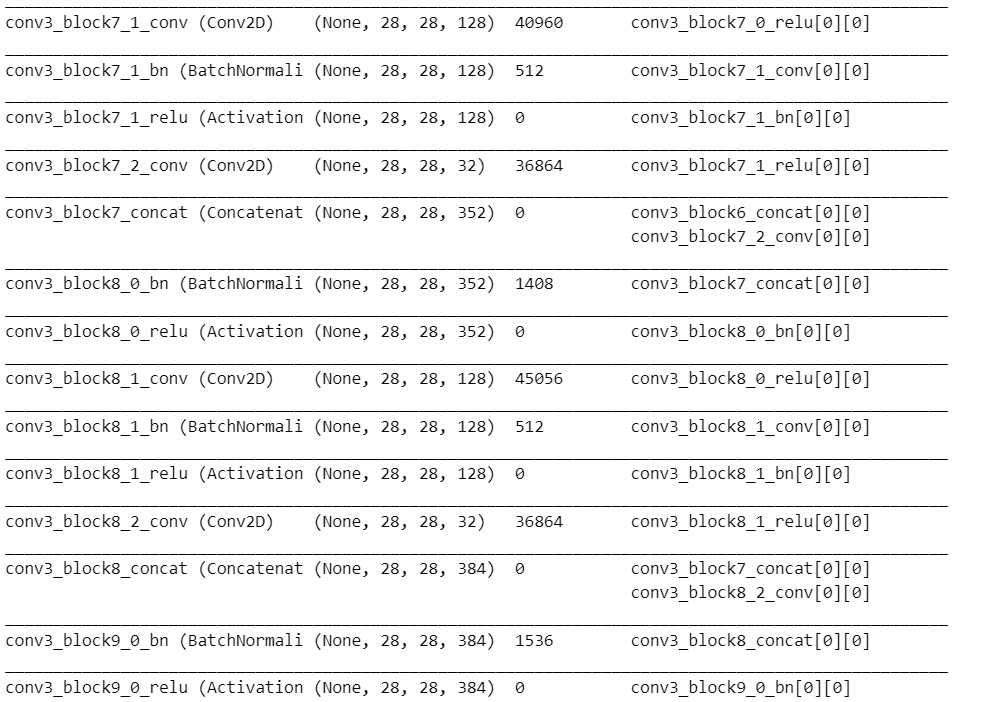


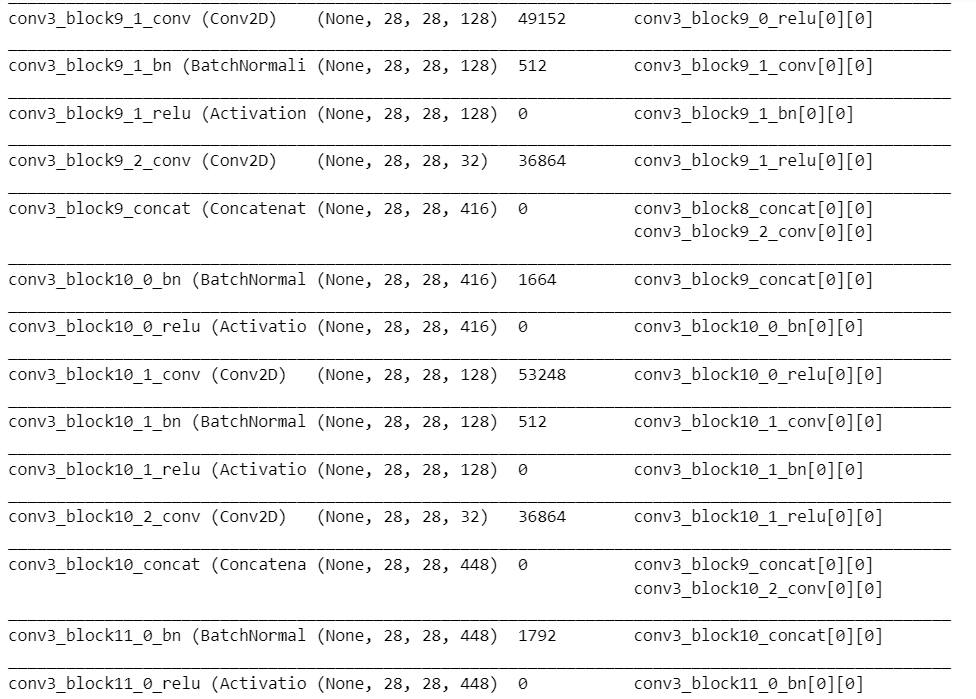


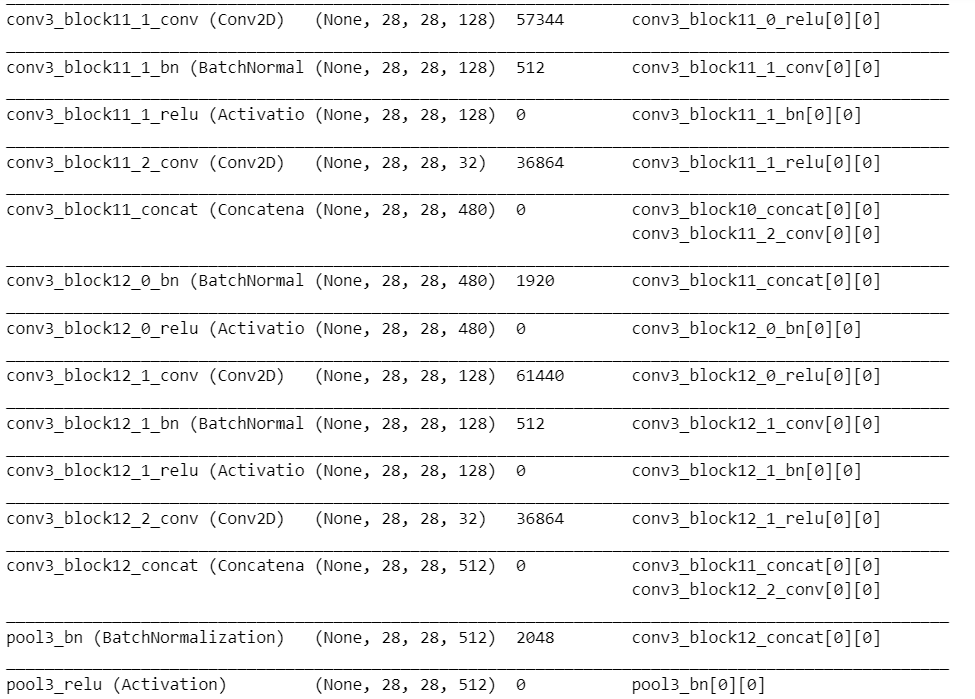


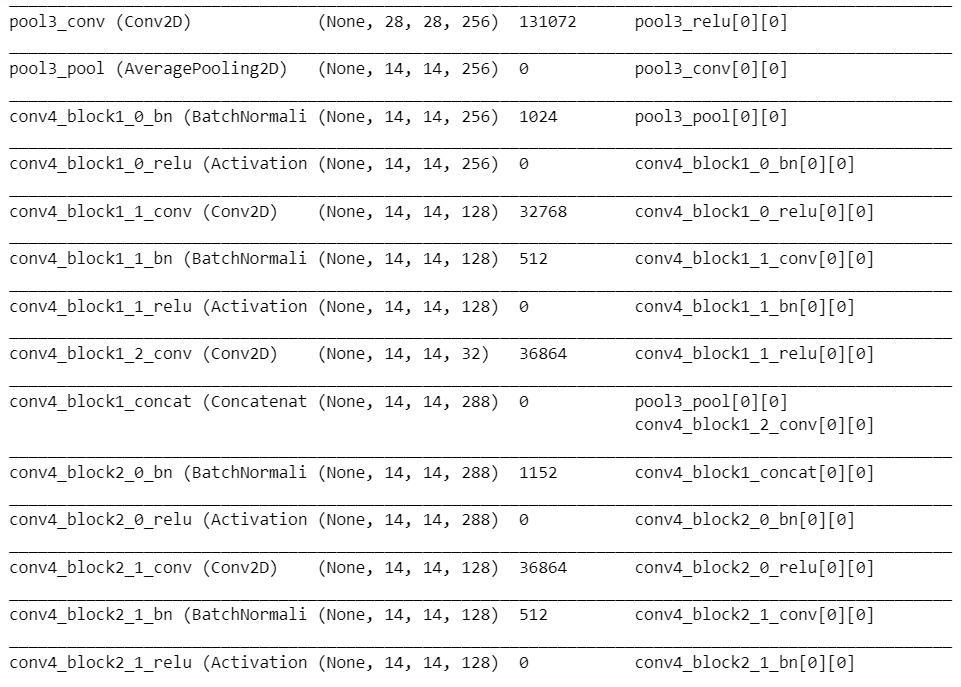


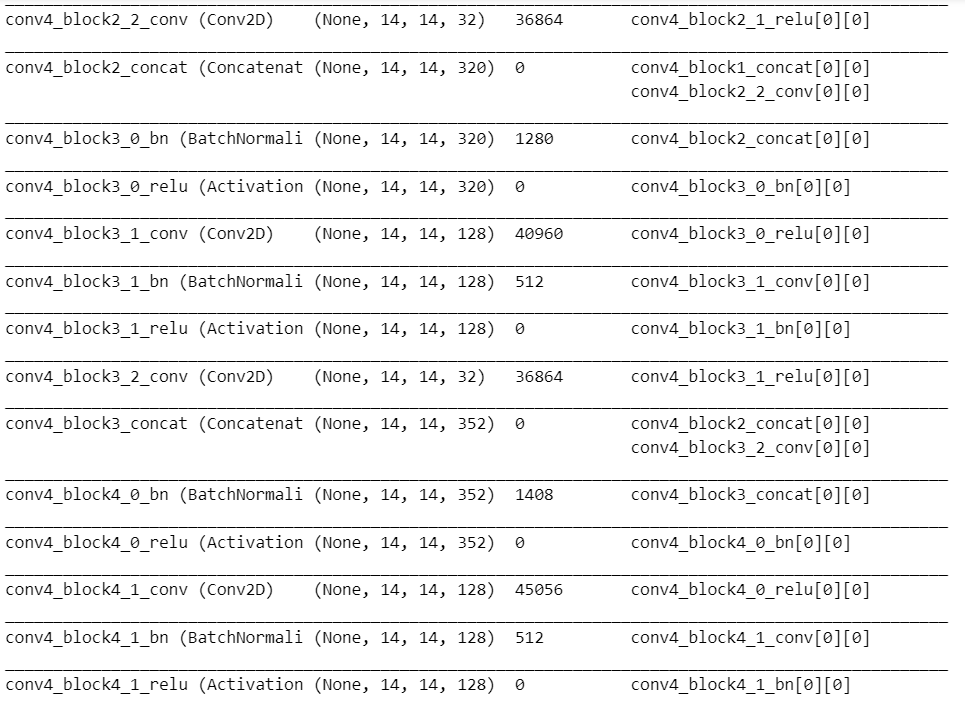


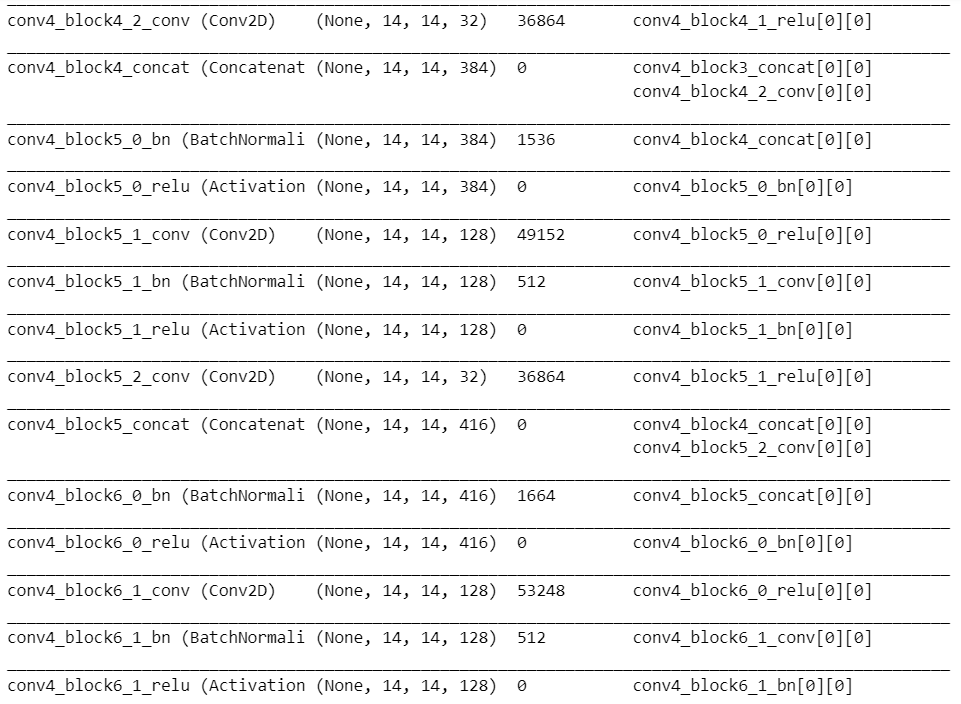


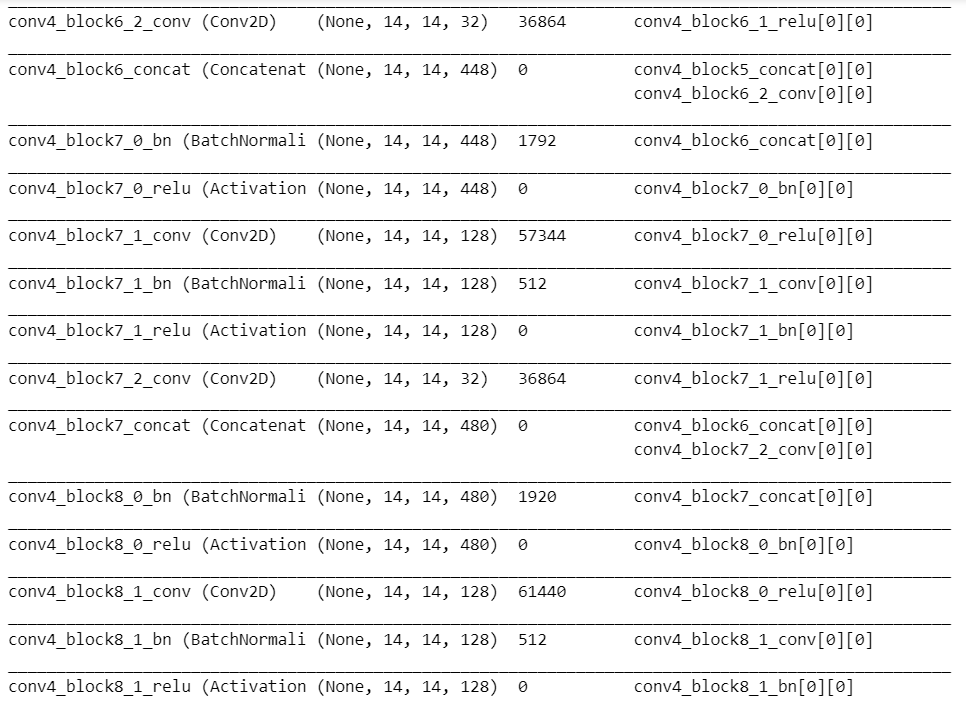


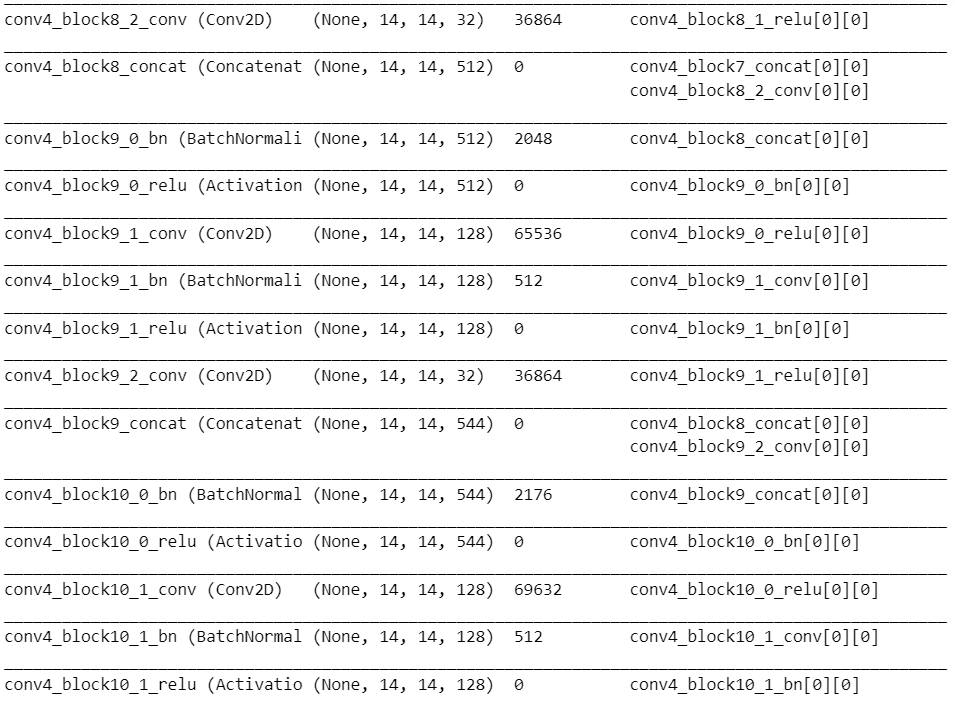


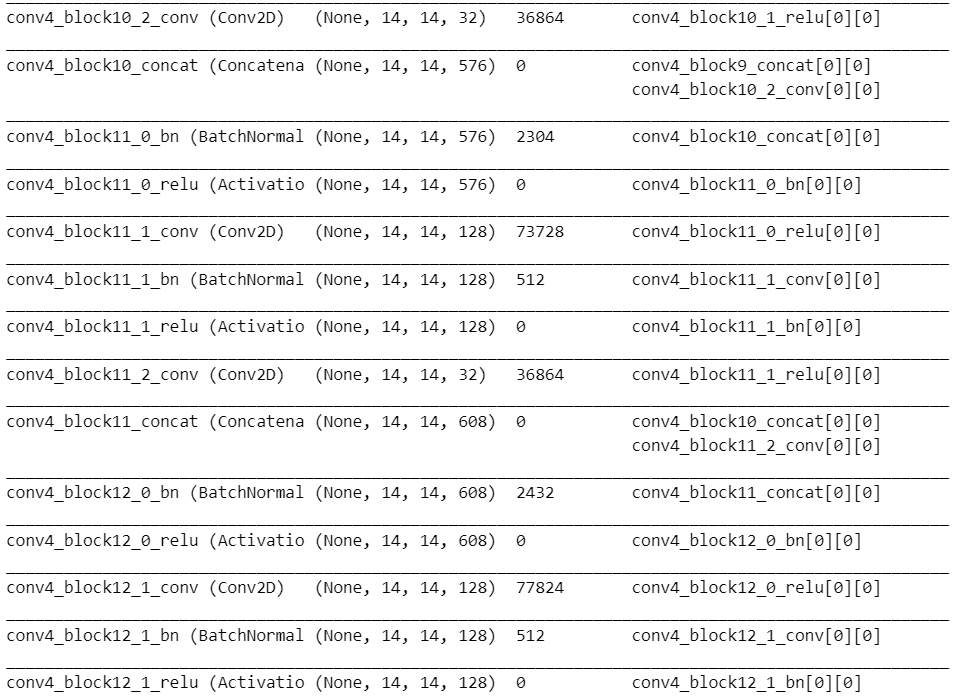


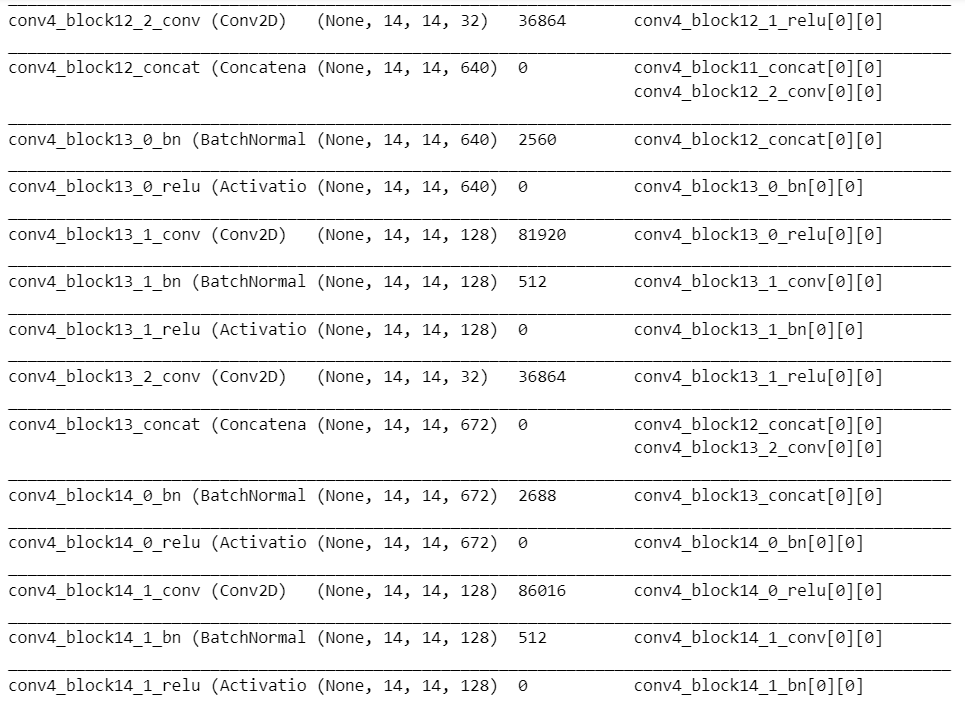


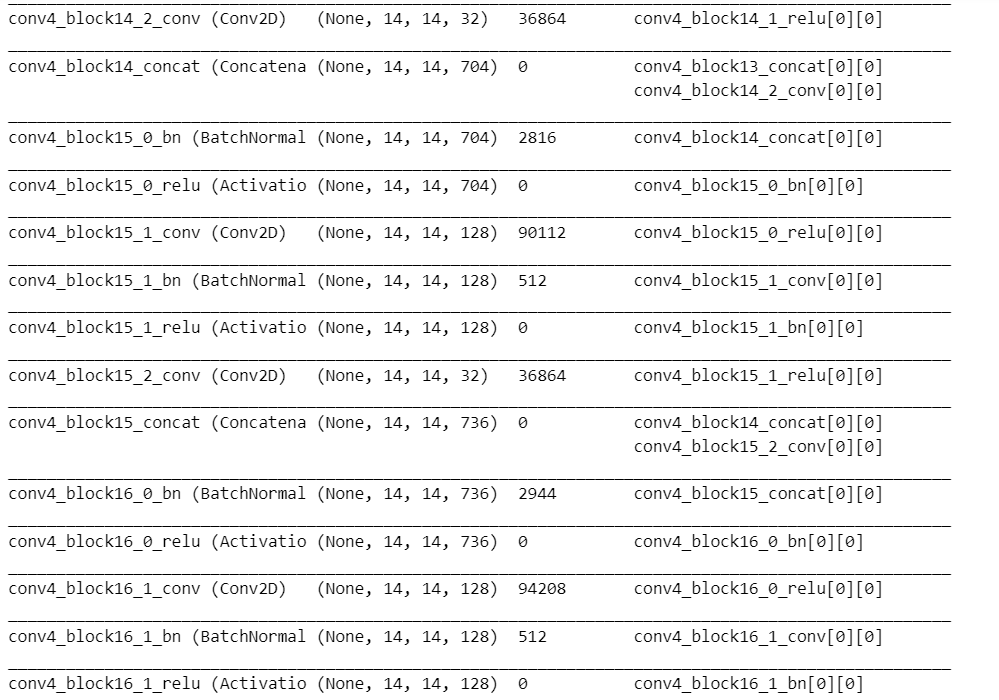


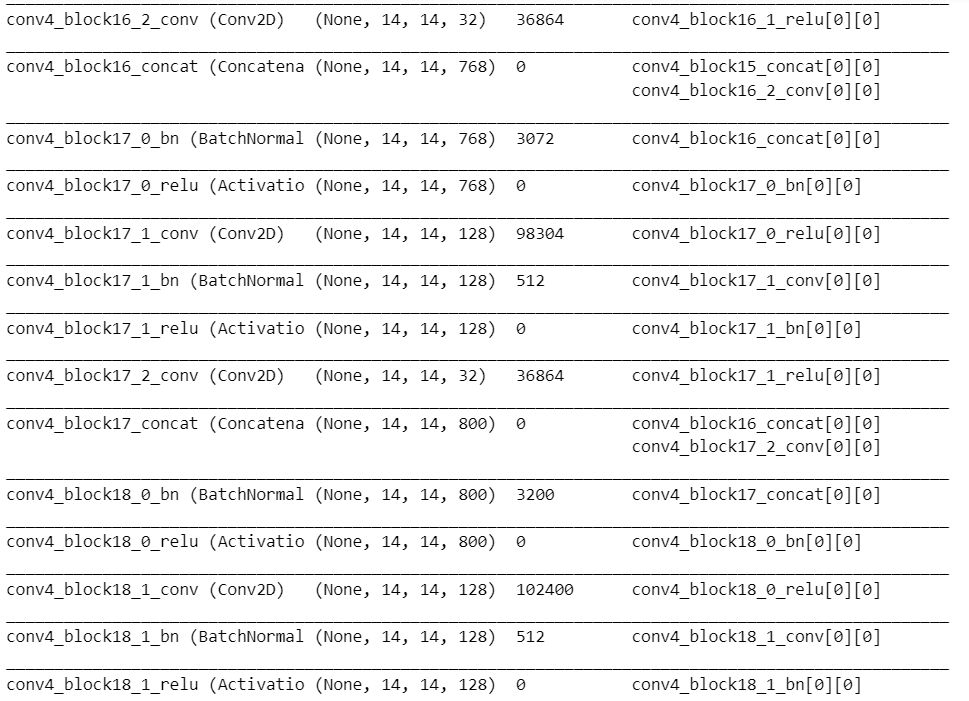


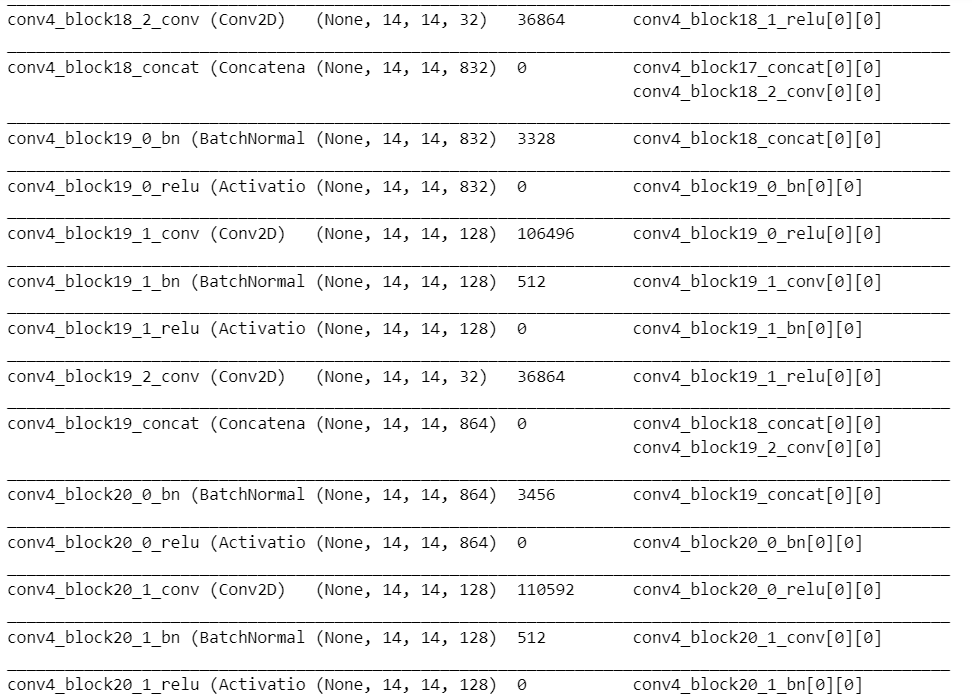


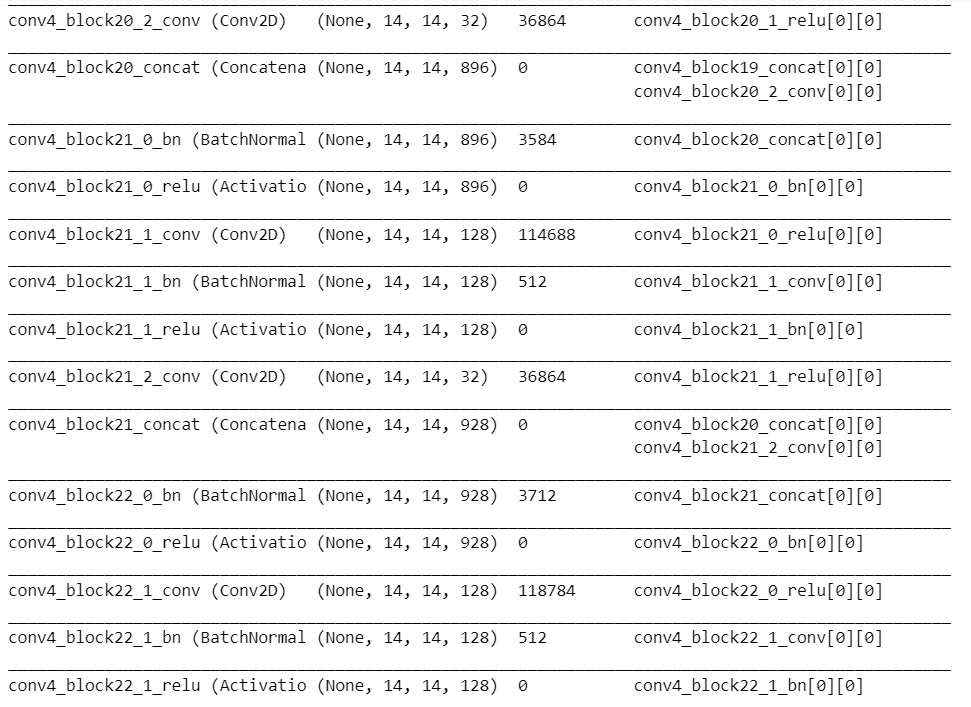


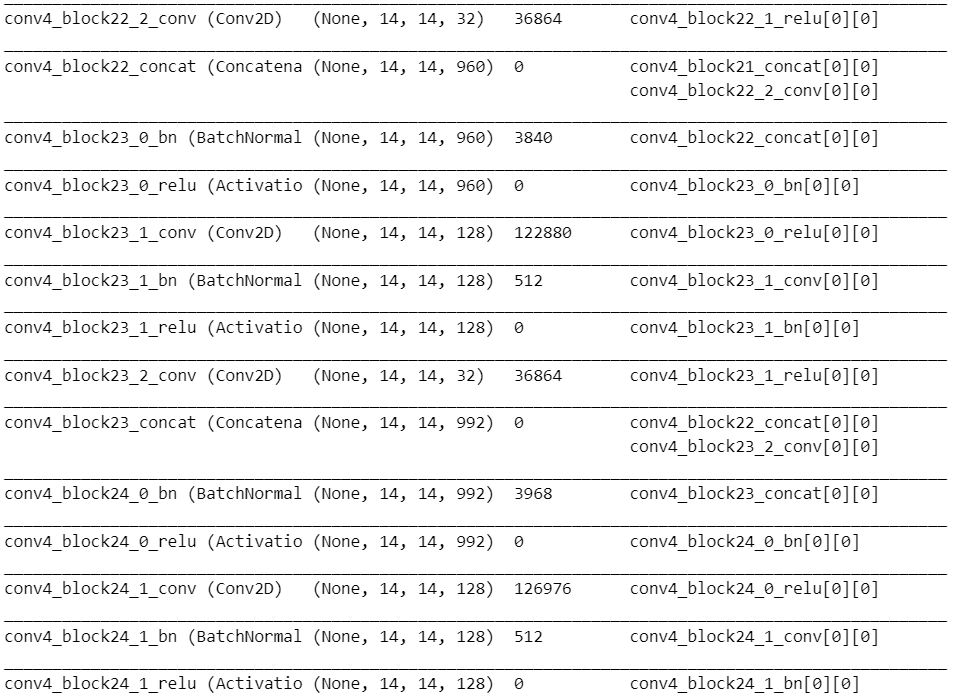


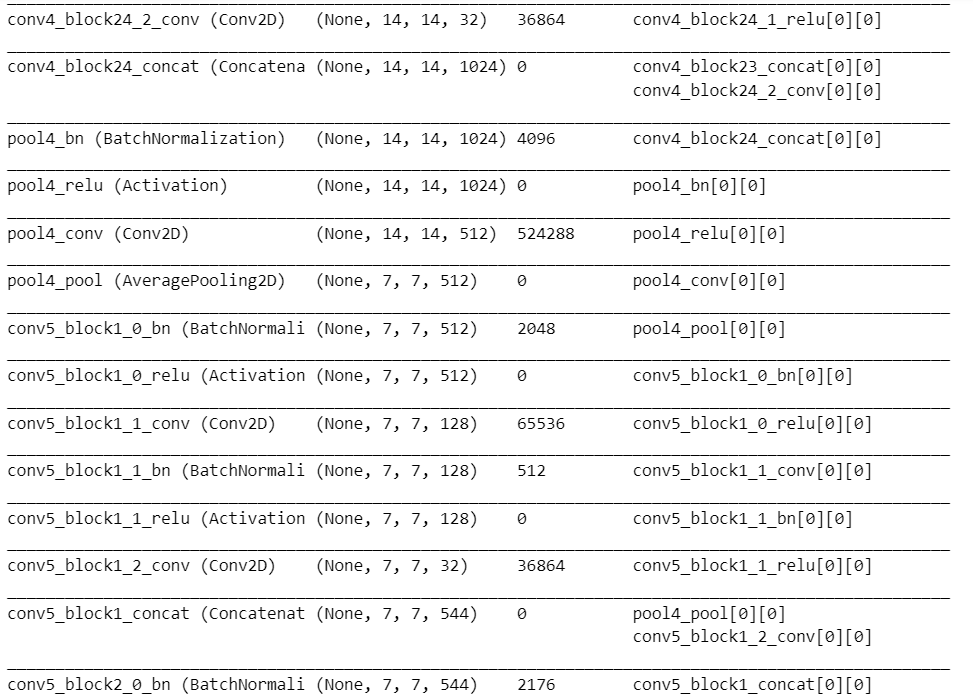


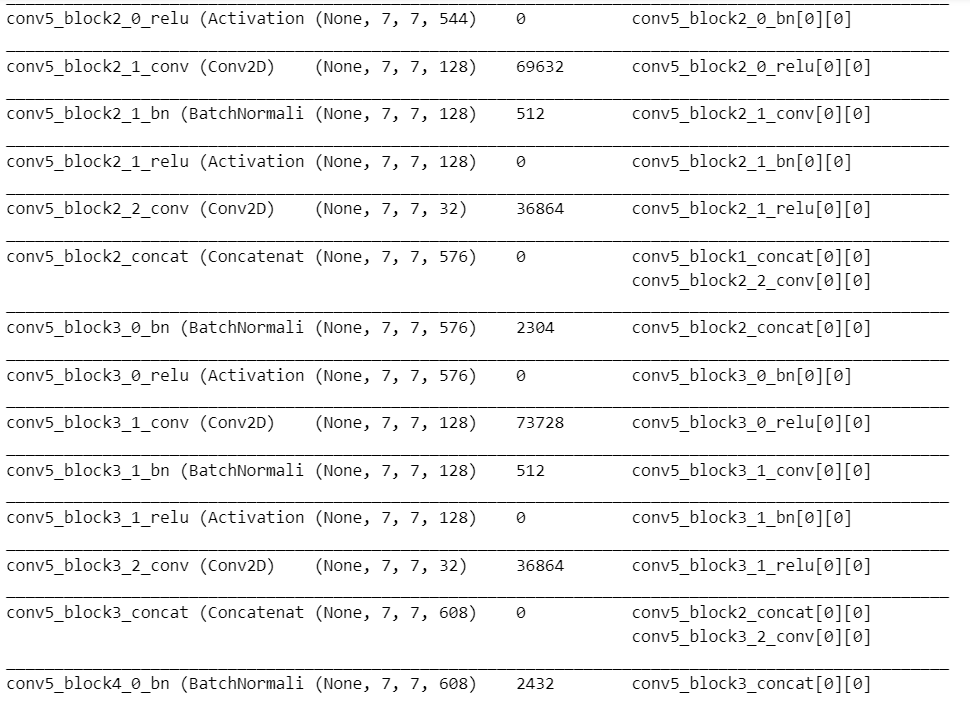


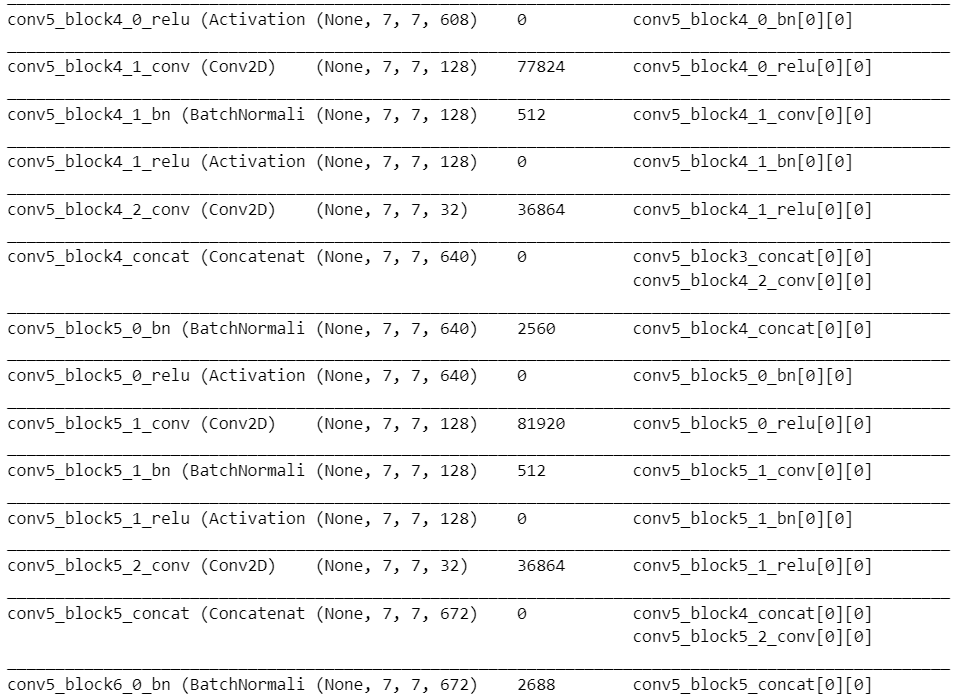


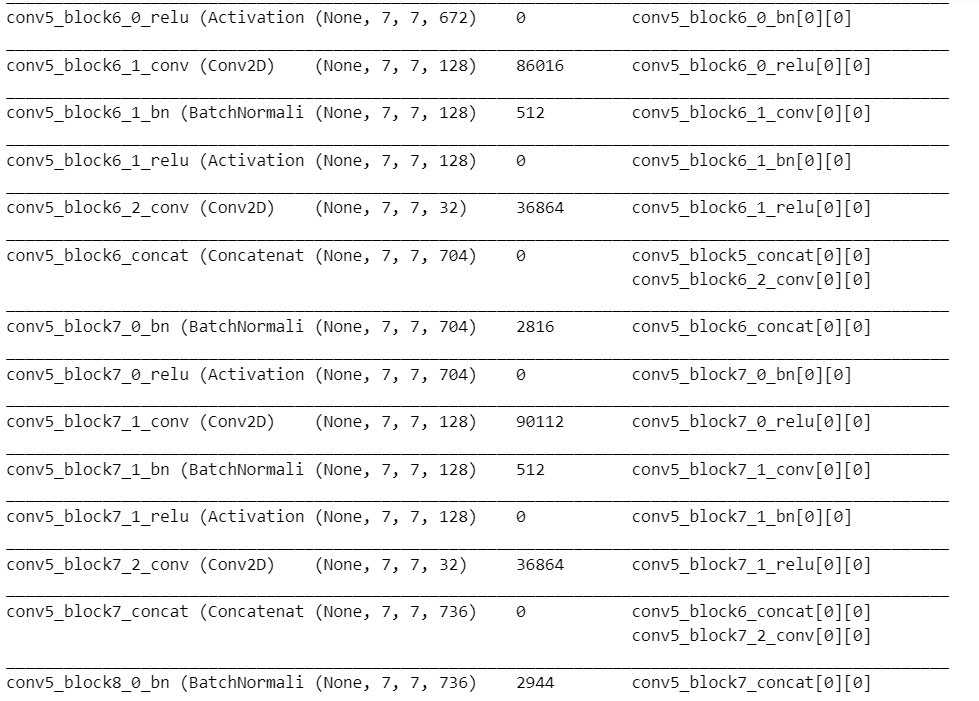


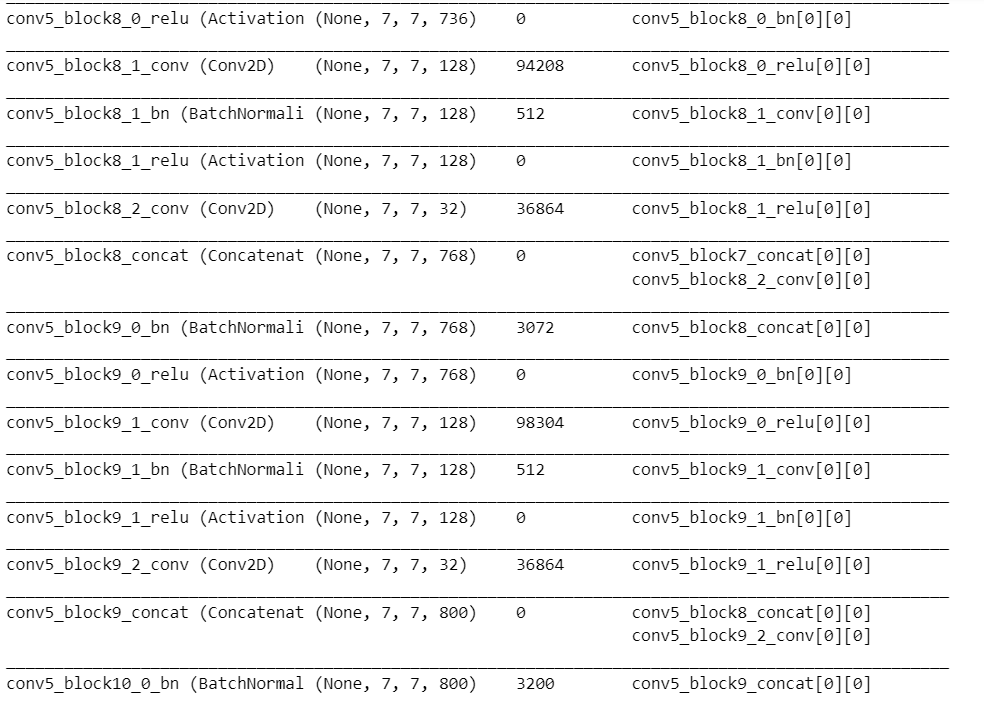


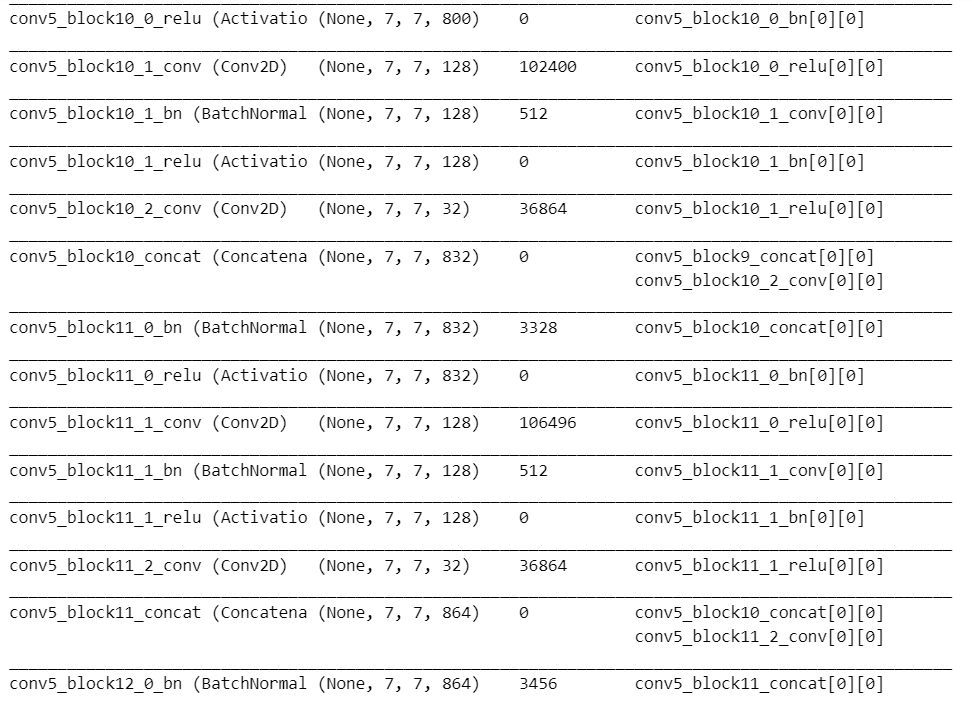


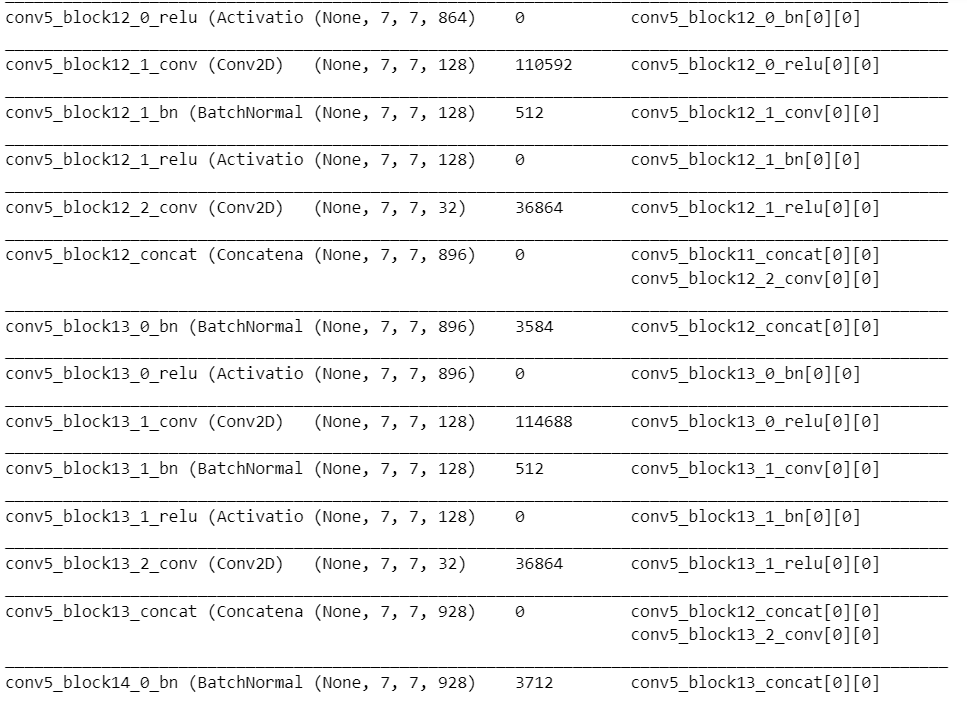


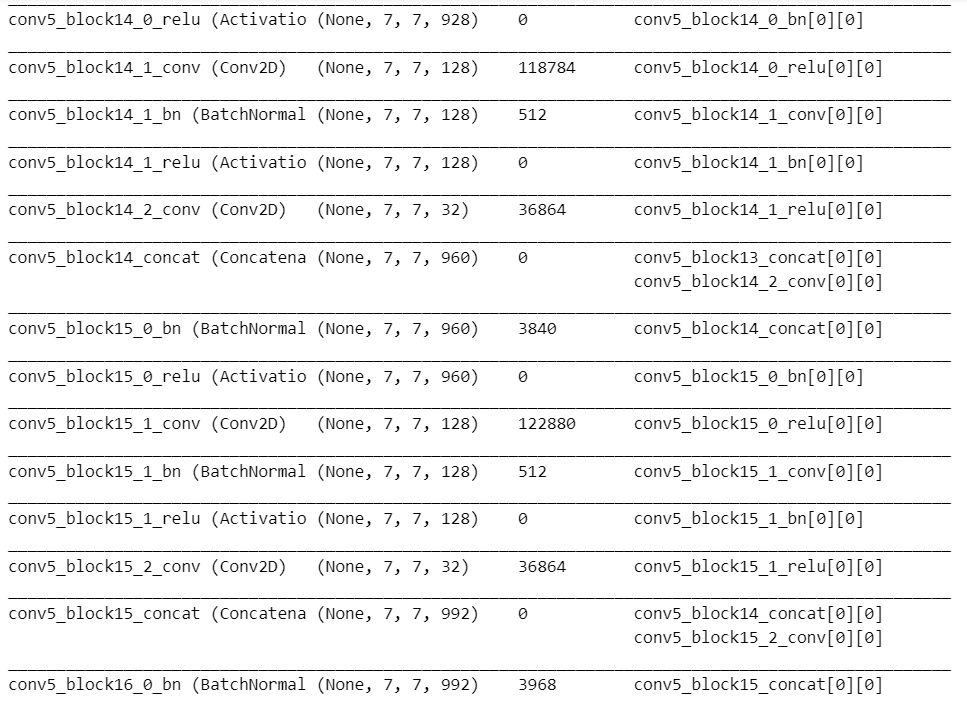


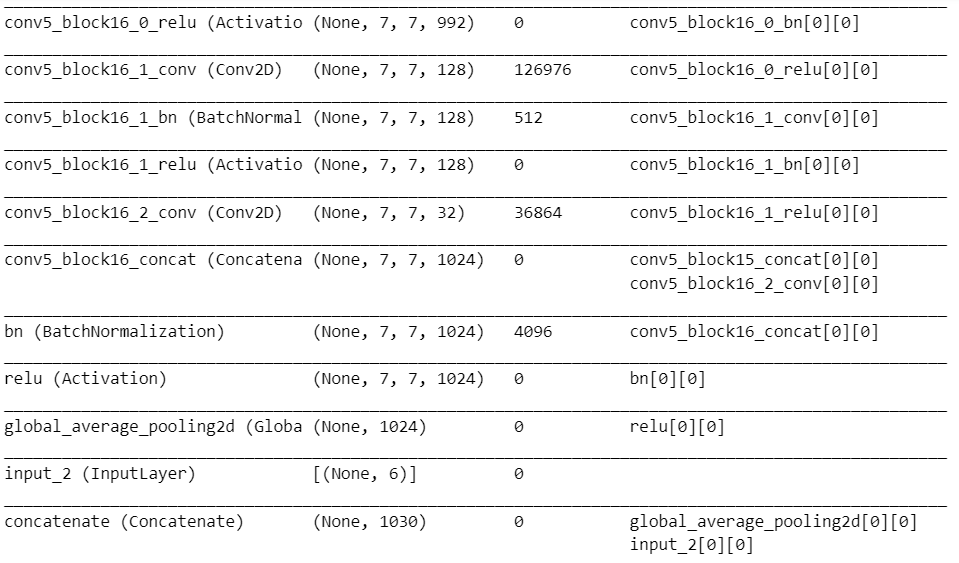


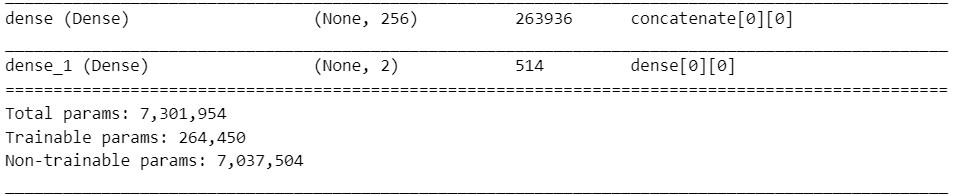










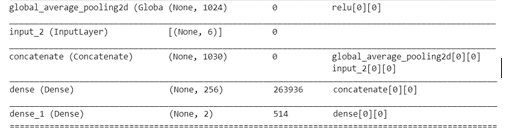


Jadi setelah input layer dari cnn, lalu input layer kedua untuk glcm feature, lalu di concatenate sebelum masuk ke fullyconnected layernya / Dense nya

Input ke cnn



Menghasilkan global average pooling, lalu di concatenate dengan input glcm



Baru masuk ke fully connected layer untuk, dan output layer dengan 2 output akhir

**Output**

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Kelas 1 | 0.93 | 0.97 | 0.95 | 39 |
| Kelas 2 | 0.97 | 0.92 | 0.95 | 39 |
|  |  |  |  |  |
| accuracy |  |  | 0.95 | 78 |
| macro avg | 0.95 | 0.95 | 0.95 | 78 |
| weighted avg | 0.95 | 0.95 | 0.95 | 78 |

Confusion Matrix:

[[38 1]

[ 1 36]]

Pada tahap ini, model dilatih menggunakan data pelatihan dan dievaluasi menggunakan data pengujian. Proses ini melibatkan:

1. Pelatihan: Model dilatih selama 10 epoch dengan batch size 16 menggunakan fungsi train\_model

2. Evaluasi: Performa model dievaluasi pada data pengujian menggunakan fungsi evaluate\_model

Hasil evaluasi mencakup:

* Classification Report: Menyediakan metrics precision, recall, dan F1-score untuk setiap kelas
* Confusion Matrix: Menunjukkan jumlah prediksi benar dan salah untuk setiap kelas