COMPUTER VISION – CONVOLUTION NEURAL NETWORK

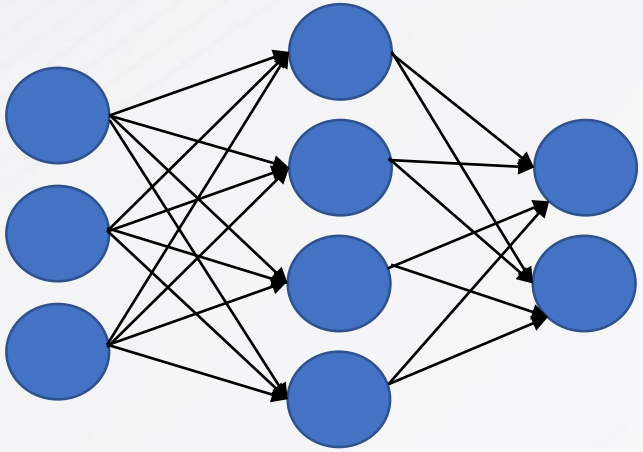
Prof. Dr.Eng. Fitri Utaminingrum, S.T., M.T.

brone.ub.ac.id

CNN - PENDAHULUAN

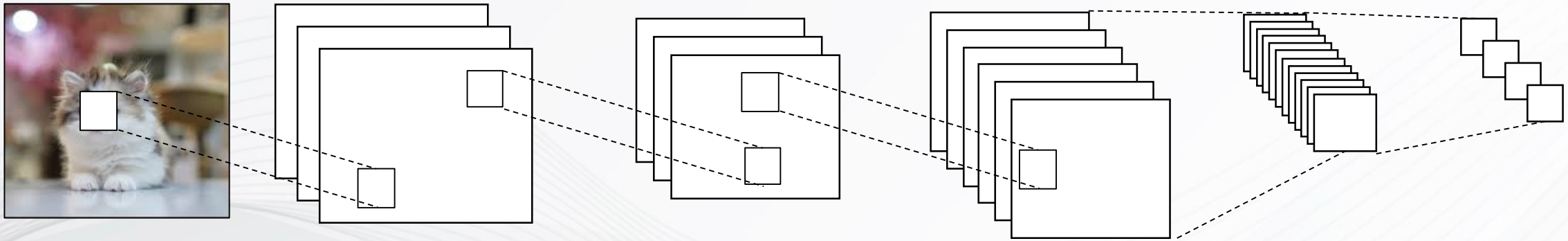
- CNN (Convolutional Neural Network) merupakan jenis *neural network* atau jaringan saraf tiruan yang memanfaatkan operasi konvolusi untuk melakukan pengenalan pola dan ekstraksi fitur secara umum dari data yang bersifat spasial
- Citra sebagai salah satu data yang bersifat spasial, CNN menjadi salah satu metode yang paling sering digunakan untuk pengenalan citra.
- CNN pertama kali dikembangkan pada tahun 1998 pada arsitektur LeNet, namun penggunaan CNN baru popular pada tahun 2012 bertepatan dengan munculnya arsitektur AlexNet dan semakin terjangkaunya komputasi GPU

CNN - PENDAHULUAN



ANN

Jaringan sarat tiruan konvensional memanfaatkan *fully-connected layer*, sedangkan pada CNN digunakan *convolution layer* yang melakukan konvolusi pada sepasang kernel/filter.

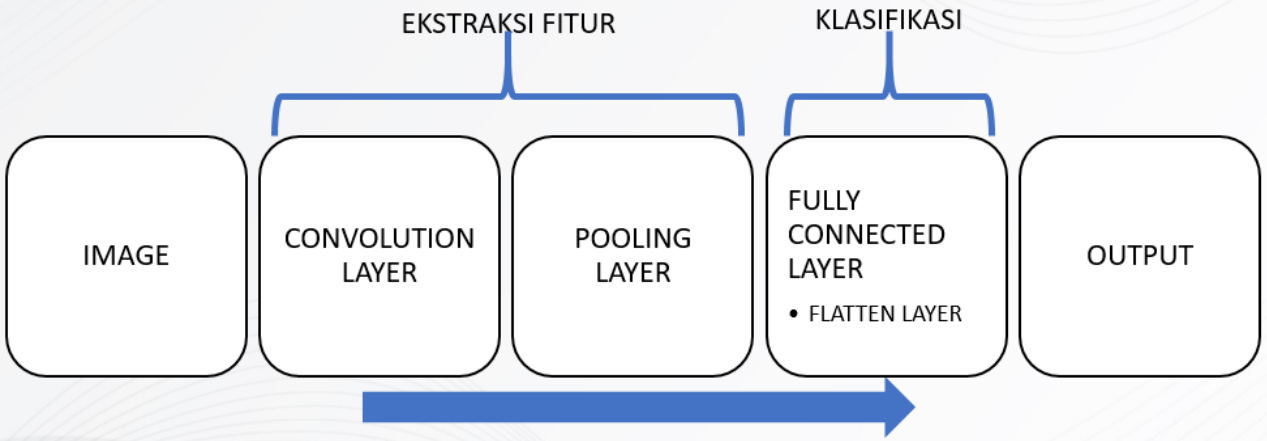


CNN

ARSITEKTUR

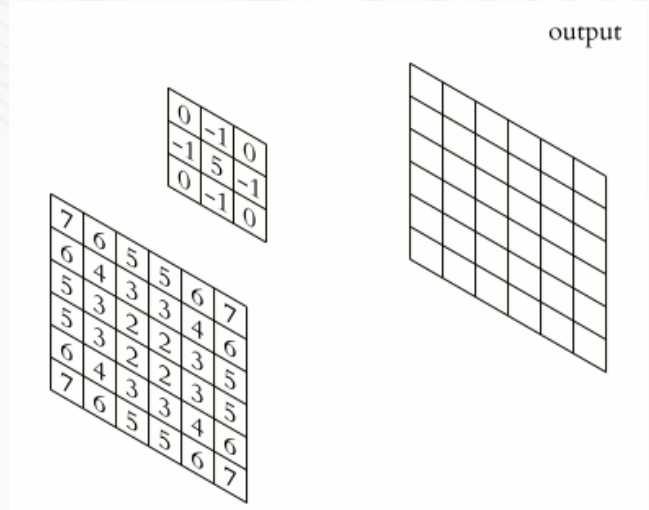
Dari diagram arsitektur LeNet dan AlexNet berikut, dapat dilihat terdapat beberapa komponen utama yang Menyusun CNN seperti :

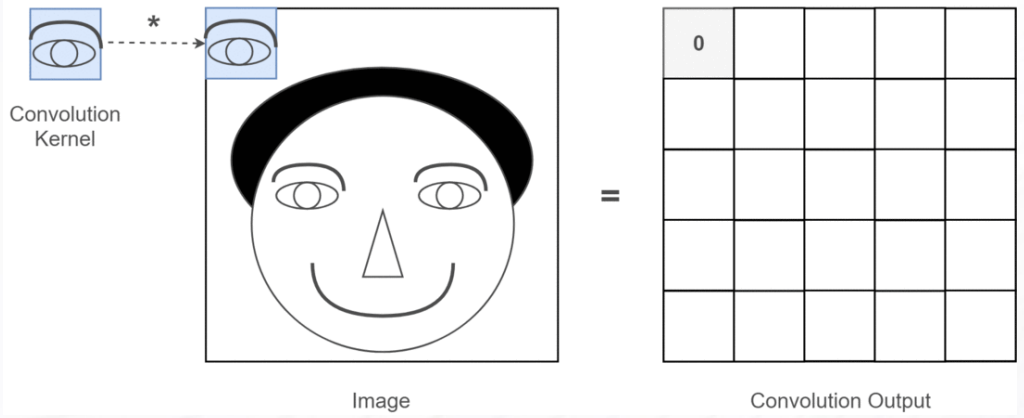
1. Convolution Layer
2. Pooling Layer
3. Dense (Fully-Connected) Layer



KONVOLUSI

- Konvolusi yang dilakukan pada layer konvolusi merupakan operasi konvolusi yang sama pada operasi konvolusi untuk deteksi tepi, filtering, dll.
- Konvolusi yang dilakukan menggunakan sepasang filter/kernel/bobot yang telah di-*train* pada data latih
- Layer ini bersamaan dengan pooling, digunakan untuk melakukan ekstraksi fitur.
- Operasi konvolusi dapat dilakukan dengan mengambil dot-product dari setiap posisi kernel pada citra asli.
- Kernel akan menghasilkan nilai *output* yang tinggi apabila terdapat sebuah pola dalam citra yang mirip dengan nilai kernel

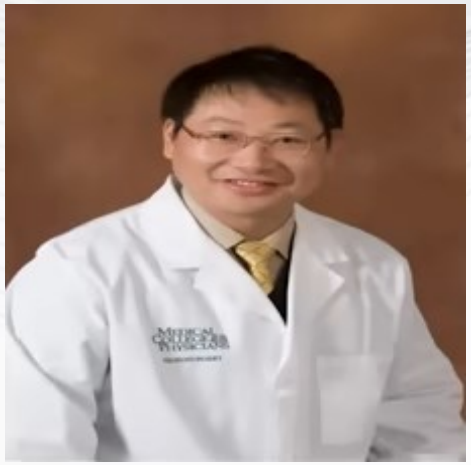




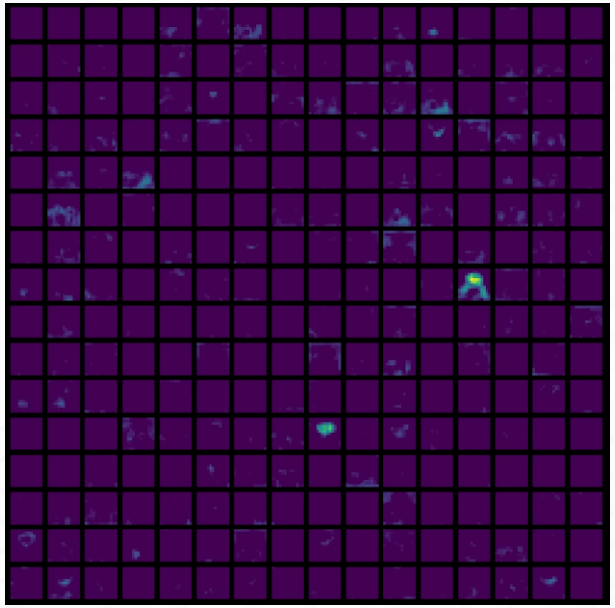
KONVOLUSI

- Dengan melakukan konvolusi berulang dan menggunakan banyak kernel, CNN dapat melakukan ekstraksi fitur tingkat tinggi yang berbeda-beda agar dapat mengenali citra
- Pada contoh dapat dilihat bahwa pada AlexNet, terdapat filter yang bereaksi kuat terhadap manusia, pakaian, dan objek lainnya

Input

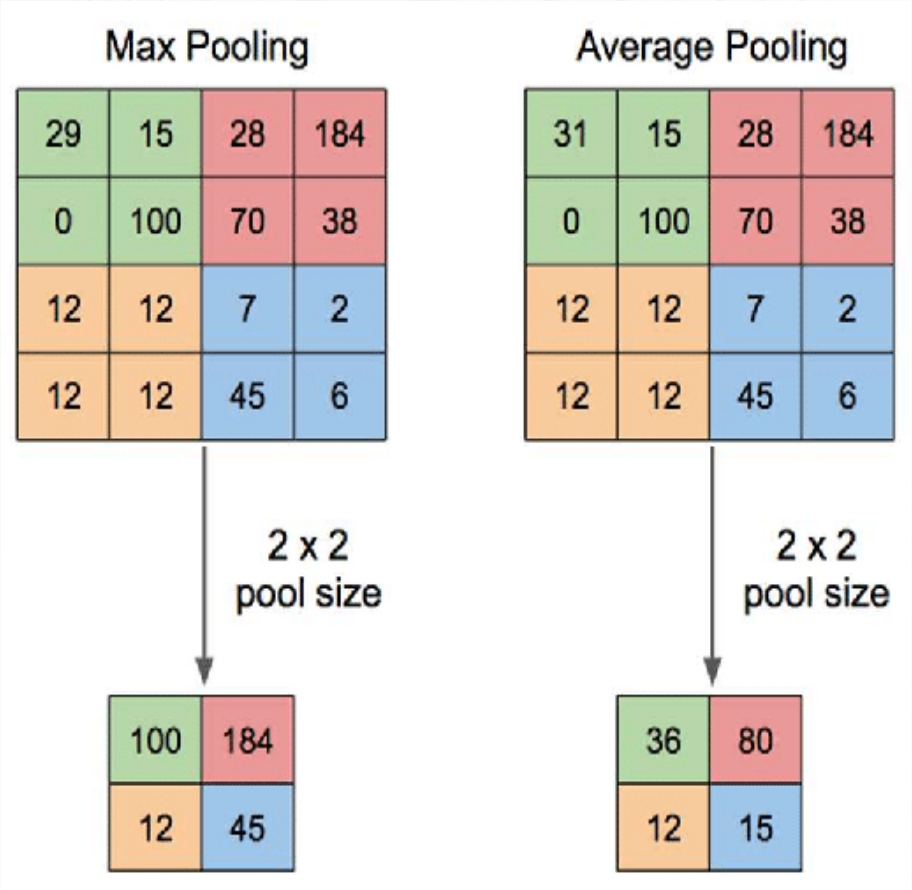


Activation Map AlexNet



POOLING

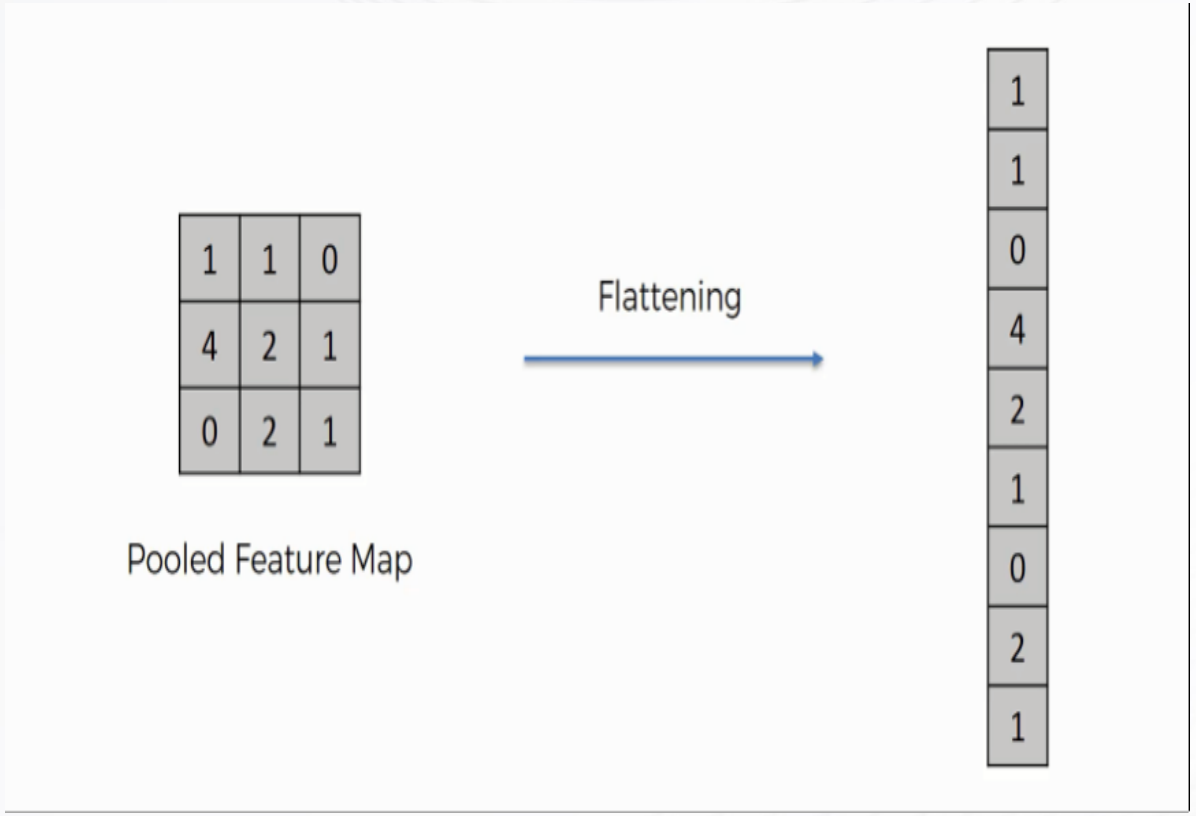
- Layer Pooling berfungsi untuk mengecilkan ukuran dimensi data citra dan membantu CNN terhadap pergeseran kecil dalam citra
- Pooling dilakukan dengan mengambil nilai maksimum (Max Pooling) atau rata-rata (Average Pooling) dari window tersebut
- Layer ini tidak mengalami proses pelatihan, melainkan dispesifikasikan dalam arsitektur
- Max Pooling akan mengambil nilai tertinggi dalam sebuah window yang akan memberikan *highlight* pada fitur yang diekstraksi dari layer konvolusi
- Average Pooling akan mengambil nilai rata-rata dalam sebuah window. Pendekatan ini tidak akan memberikan *highlight* pada fitur, namun dengan menggunakan rata-rata, keseluruhan fitur akan tetap tergambarkan pada hasil pooling



FLATTEN

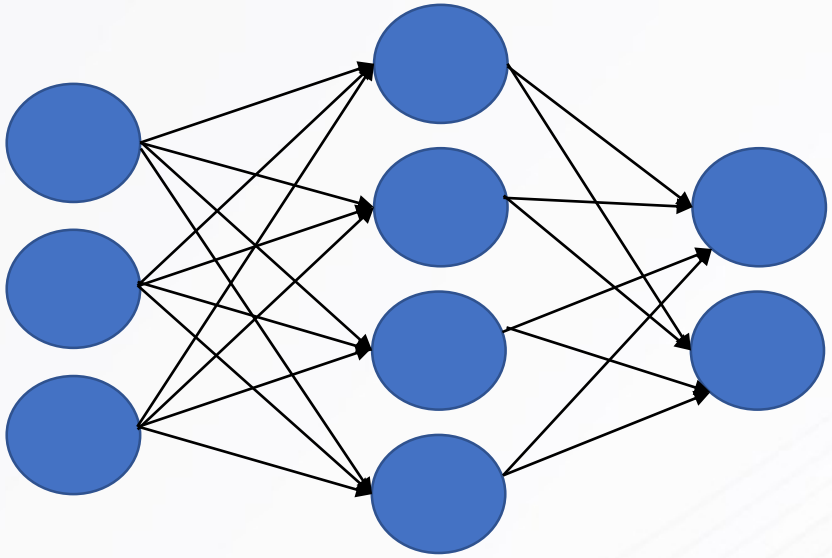
- Fitur yang didapatkan dari hasil layer konvolusi dan pooling sebelumnya akan melalui layer flatten data agar dapat dimasukkan kedalam dense layer

- Flatten layer bekerja dengan meratakan data berdimensi tinggi menjadi satu dimensi secara berurutan



DENSE LAYER

- Dense Layer merupakan Fully-Connected layer yang digunakan untuk melakukan klasifikasi
- Bekerja sebagaimana FCNN/MLP lainnya dengan input berupa hasil dari layer flatten
- Merupakan bagian terberat pada CNN konvensional (AlexNet, LeNet, VGG)

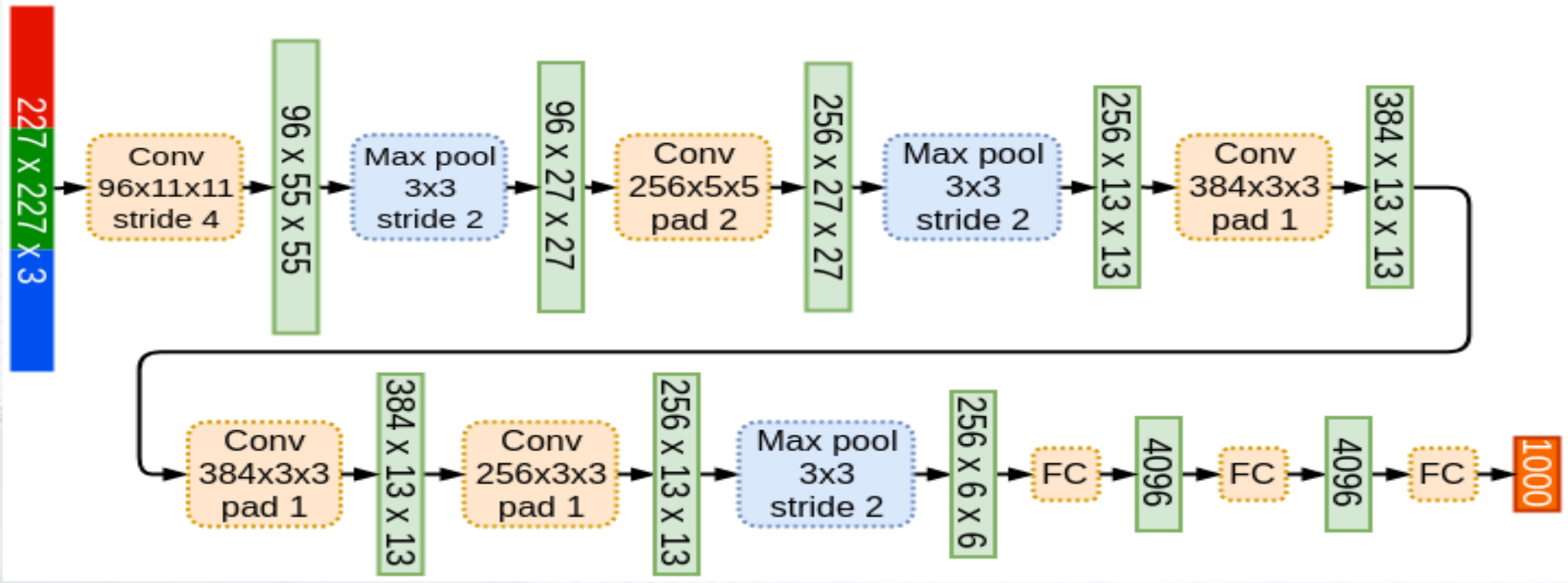


IMPLEMENTASI CNN

- CNN umumnya diimplementasikan menggunakan library Machine Learning untuk mempermudah pembuatan model dan mempercepat proses komputasi dengan menggunakan komputasi GPU
- Dua library ML populer yang dapat digunakan untuk mengimplementasikan CNN adalah **Tensorflow** dan **PyTorch**

ALEXNET

- Arsitektur AlexNet terdiri dari 5 layer konvolusi dengan 3 max pooling yang kemudian diteruskan dengan 3 layer fully-connected.
- AlexNet dapat dibagi menjadi 2 bagian. Yaitu bagian backbone (Konvolusi dan Pooling) dan head (Fully-Connected)



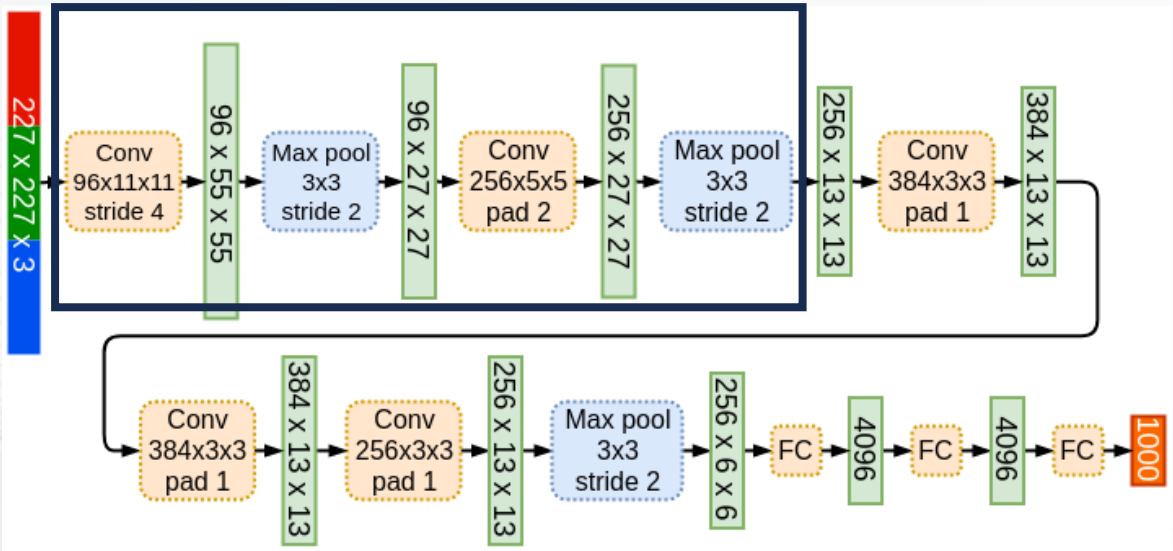
IMPLEMENTASI ALEXNET

- AlexNet dapat diimplementasikan menggunakan berbagai library seperti Tensorflow, PyTorch, JAX, ataupun library lainnya.
- Dalam contoh implementasi ini akan digunakan Tensorflow dengan menggunakan model Sequential

```
model = Sequential([
    Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4), activation='relu', input_shape=(227, 227, 3)),
    MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

    Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

    Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
    Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
    Conv2D(filters=256, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

    Flatten(),


    Dense(units=4096, activation='relu'),
    Dense(units=4096, activation='relu'),
    Dense(units=1000, activation='softmax')
])
```

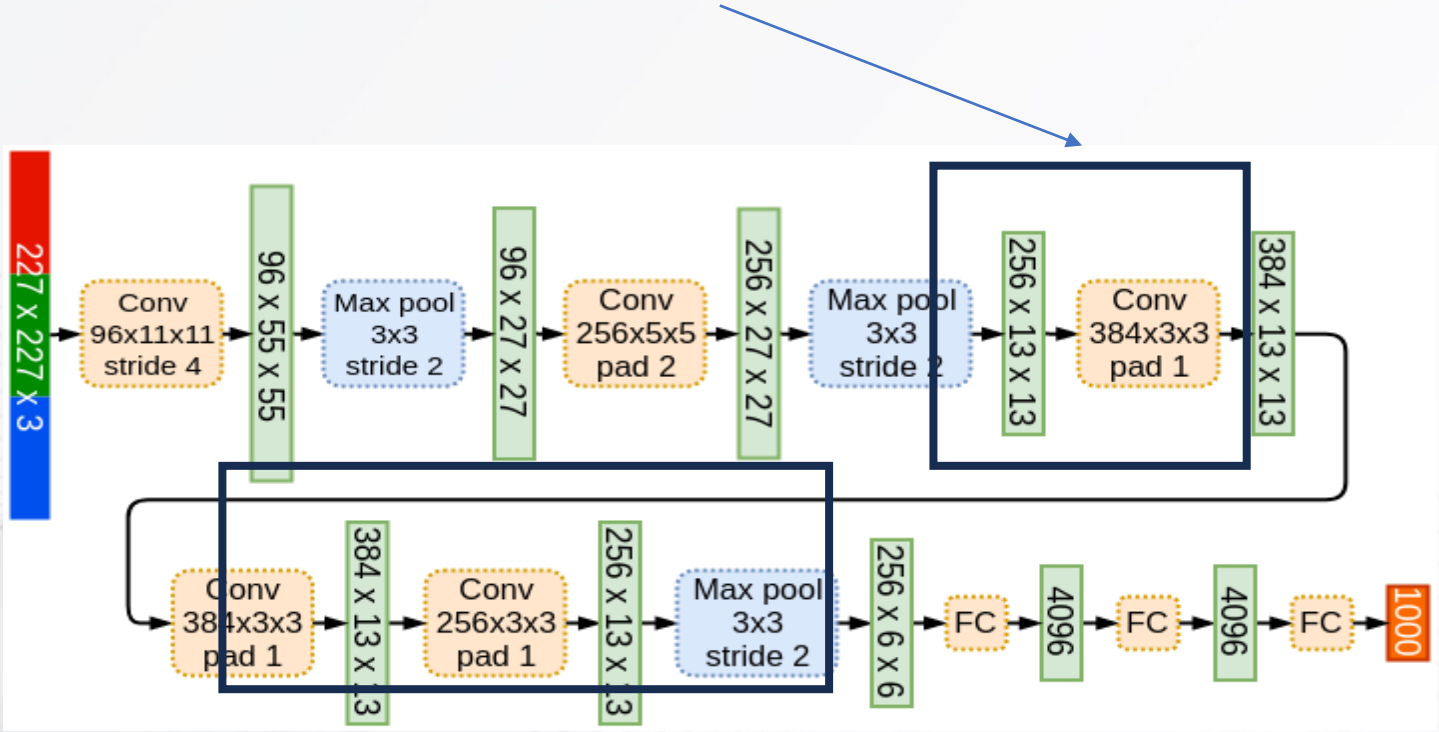
IMPLEMENTASI ALEXNET

```
Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4), activation='relu', input_shape=(227, 227, 3)),
MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'),
MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
```

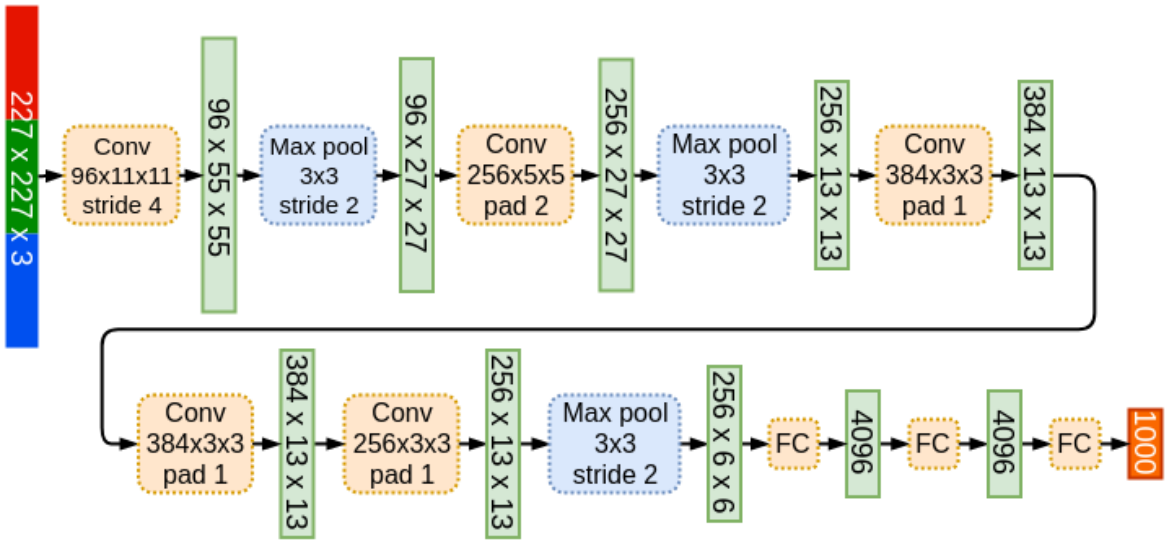


IMPLEMENTASI ALEXNET

```
Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
Conv2D(filters=256, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
```



IMPLEMENTASI ALEXNET




```
Flatten(),

Dense(units=4096, activation='relu'),
Dense(units=4096, activation='relu'),
Dense(units=1000, activation='softmax')
```

IMPLEMENTASI ALEXNET

- Model AlexNet yang telah diimplementasikan kemudian dapat dilatih untuk melakukan klasifikasi citra

- AlexNet memiliki jumlah 62.378.344 parameter dengan sekitar 7 juta parameter pada bagian backbone dan 55 juta pada bagian head.

```
=================================================================
Total params: 62378344 (237.95 MB)
Trainable params: 62378344 (237.95 MB)
Non-trainable params: 0 (0.00 Byte)
```