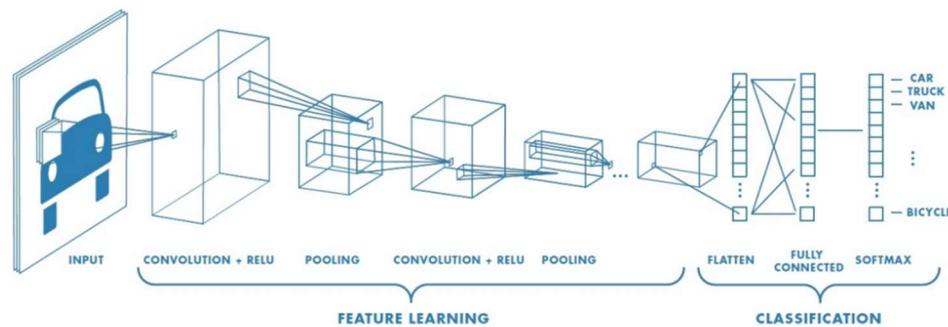


dikomputasi, sedangkan untuk *fully connected* berguna untuk pengklasifikasi dari fitur ekstraksi [22]. Untuk *fully connected layers* biasanya digunakan bersamaan dengan fungsi aktivasi *Softmax* untuk tujuan klasifikasi. Berbagai macam arsitektur CNN yang bisa digunakan langsung tanpa harus membuat dari awal untuk bagian *feature learning* dan klasifikasinya, contoh dari arsitektur tersebut seperti AlexNet, OverFeat, GoogleNet, dan lain-lain [22].

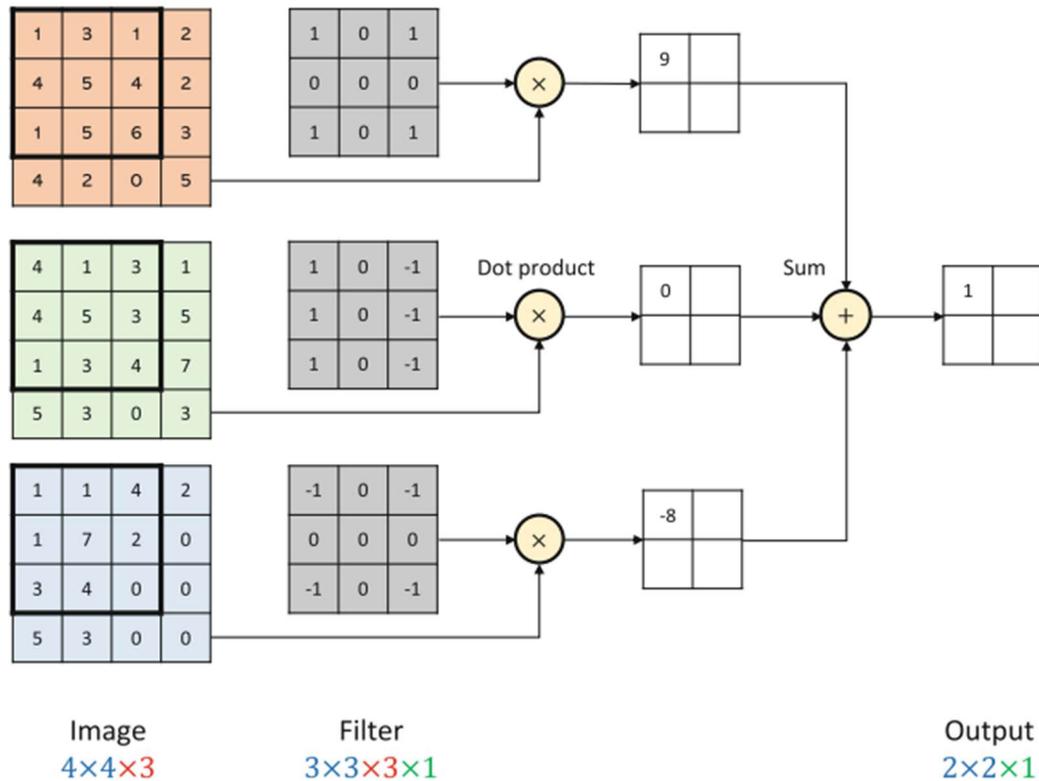
Perbedaan yang bisa dilihat dari ANN dan CNN adalah perlu diketahui jika CNN adalah arsitektur lanjutan dari ANN, dimana CNN memiliki *convolutional layers*, *stride*, *padding*, *pooling layers*, dan *full-connected layers* [5]. Biasanya di dalam arsitektur CNN, setiap *convolution layers* terdapat ReLU dan *Pooling layer* secara berulang-ulang seperti yang direpresentasikan pada Gambar 2.8, lalu diikuti oleh lapisan untuk klasifikasi seperti *flatten*, *fully connected*, dan *activation layer* seperti *softmax*.



Gambar 2.8 Gambaran arsitektur CNN.

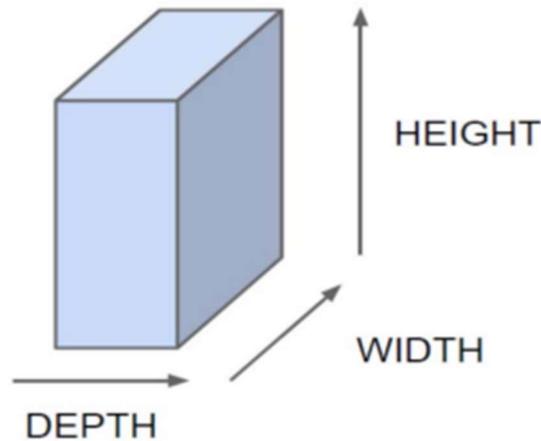
2.4.1 Convolutional Layer

Convolutional layer adalah lapisan pertama yang bertugas melakukan konvolusi pada citra data di *layer input* [23]. Pada *convolutional layer* proses konvolusi menggunakan filter bisa dilakukan ke citra gambar (dengan panjang *pixel*, tinggi *pixel*, dan juga kedalaman tiga *pixel* berupa RGB *channel*) atau sebuah video (video *grayscale* di mana memiliki resolusi tinggi dan lebar, sedangkan kedalamannya berupa *frames*) [5]. Jumlah dari kanal dan ukuran filter bisa disesuaikan dengan jumlah dari *input* citra data.



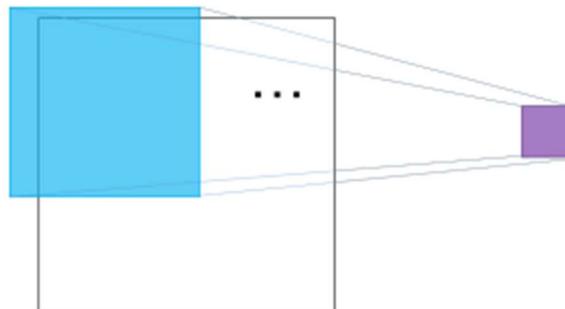
Gambar 2.9 Proses pada *convolutional layer* dengan tiga channel (RGB).

Kita ambil contoh dengan jaringan yang mendapatkan nilai *pixels* mentah sebagai *input*. Kemudian, untuk menghubungkan *input layers* ke satu *neuron*, akan ada $32 \times 32 \times 3$ parameter jika menggunakan dataset CIFAR-10 [5] [24]. Jika kita menambahkan satu *neuron* lagi ke dalam *hidden layer*, maka kita memerlukan $32 \times 32 \times 3$ parameter lagi, di mana akan menjadikan parameter dari kedua *neuron* tersebut menjadi $32 \times 32 \times 3 \times 2$ parameter [5]. Untuk memperjelas lagi, lebih dari 6000 parameter digunakan untuk menghubungkan *input* ke dua *nodes neurons* saja [5]. Dengan nilai parameter tersebut kita dapat membuat mesin menganalisa hingga tahapan lapisan *output*.



Gambar 2.10 Representasi *input* tiga dimensi dari CNN.

Perlu diingat jika tujuan awal CNN adalah untuk mengenali aspek informatif atau spesial yang ada pada region tertentu.



Gambar 2.11 *Sliding window*.

Untuk mengenali objek pada banyak regional maka memerlukan yang namanya *sliding window* seperti pada Gambar 2.10, dengan ilustrasi pada bagian warna biru merupakan representasi satu *window*, kemudian kotak ungu merupakan representasi dari aspek paling spesial (disebut *filter*) dari *window* tersebut. Setiap operasi pada *window* bertujuan untuk mencari aspek spesial yang ada pada *window* tersebut. Dengan kata lain kita ingin mendapatkan nilai numerik (*filter*) dari mentransformasikan suatu *window*. Kita juga bisa mendapatkan nilai d (d -channels) numerik dari mentransformasikan suatu *window*. *Window* ini kemudian digeser sebanyak T kali, sehingga mendapatkan vektor dengan panjang $d \times T$ [25].

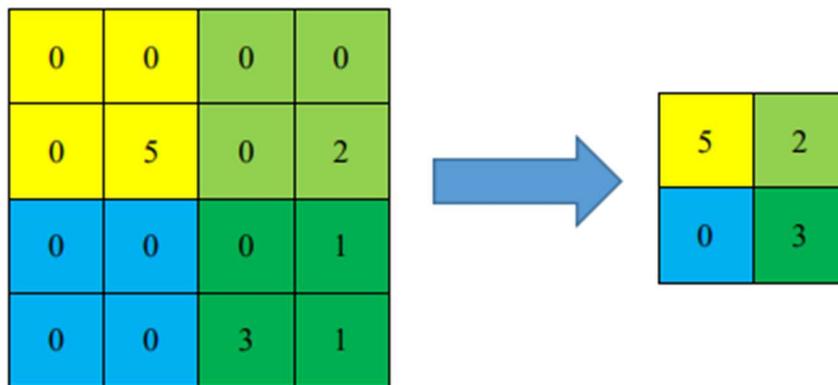
2.4.2 ReLU

ReLU berguna untuk mengubah nilai negatif menjadi nol dan nilai positif menjadi nilai tetap [20].

$$\text{ReLU } f(x) = \max(0, x) \quad (2.1)$$

2.4.3 Pooling Layer

Tujuan utama dari *pooling layer* adalah *downsampling* untuk mengurangi kompleksitas lapisan selanjutnya. Di dalam *image processing*, fitur ini berfungsi untuk mengurangi resolusi citra data [5]. Salah satu metode *pooling* yang sering digunakan adalah *max pooling* di mana cara kerja *max pooling* adalah mempartisi citra data ke *sub-region* persegi panjang dan mengembalikan nilai maksimum dari dalam *sub-region* tersebut [5]. Setelah melewati berbagai macam operasi *convolution* dan *pooling* akan dihasilkan satu vektor yang kemudian akan dilewatkan pada *fully connected layer* untuk melakukan identifikasi suatu permasalahan tertentu [25].



Gambar 2.12 Max pooling.

2.4.4 Fully Connected Layer

Fully connected layer sama seperti dengan *convolutional layer*, yang membedakannya adalah *fully connected layer* terhubung ke semua *neuron* sedangkan *convolutional layer* hanya menghubungkan beberapa saja pada *layer input* [5]. Fungsinya adalah mengubah atau melakukan *flatten feature map* agar