

CNNs (More)

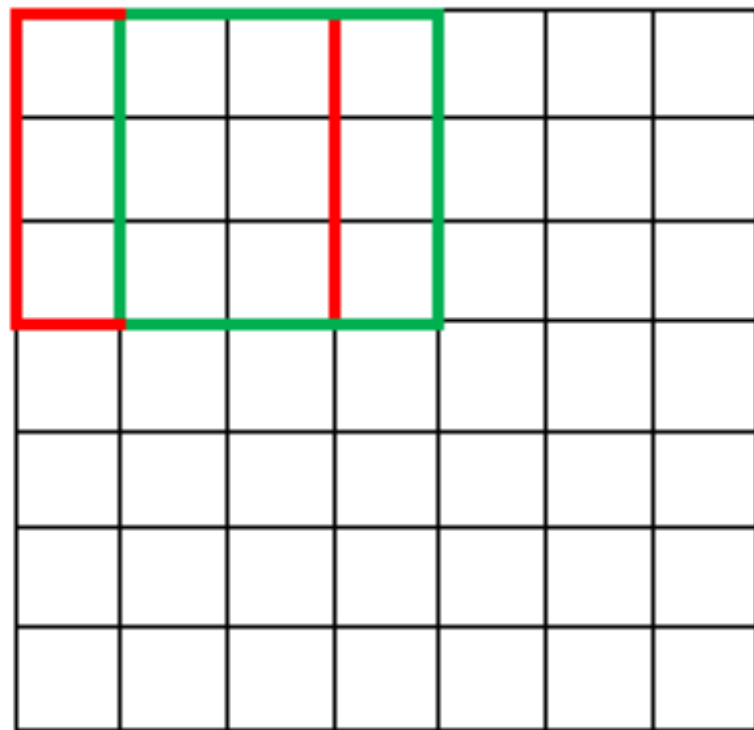
Review

- CNNs:
 - Stride & padding
 - Regularization:
 - Dropout
 - Batch normalization
- Fine-tuning

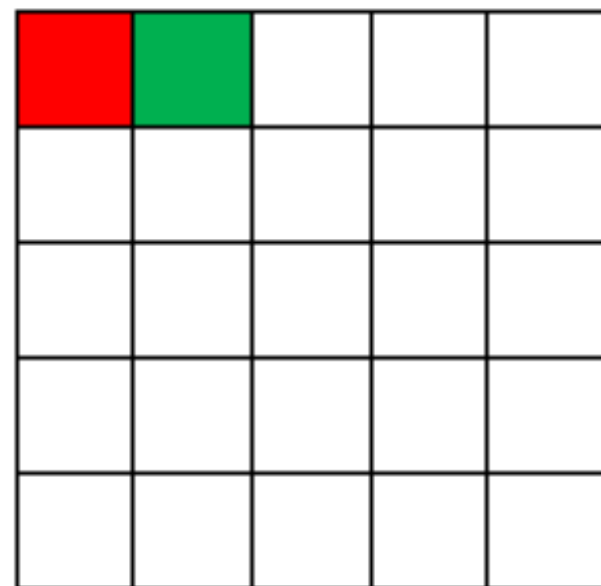
Stride

Stride (1, 1)

7 x 7 Input Volume

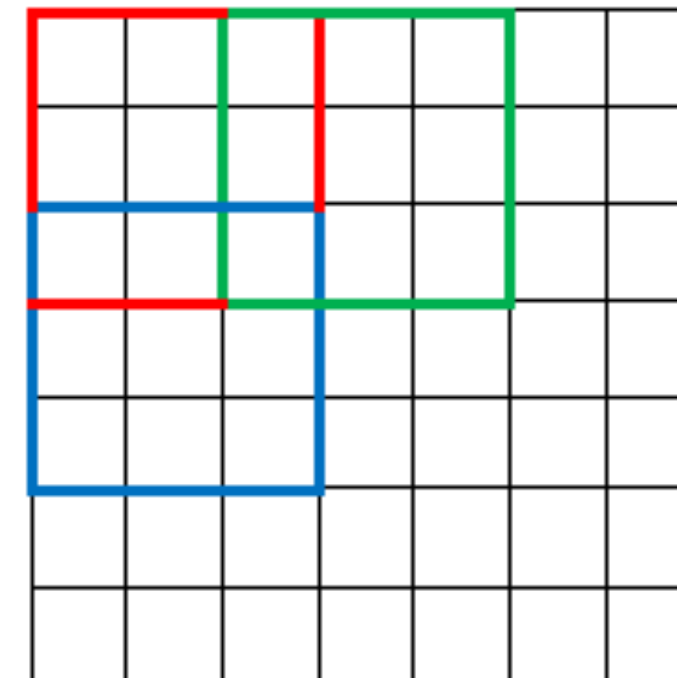


5 x 5 Output Volume

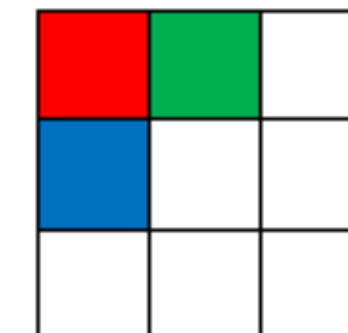


Stride (2, 2)

7 x 7 Input Volume



3 x 3 Output Volume

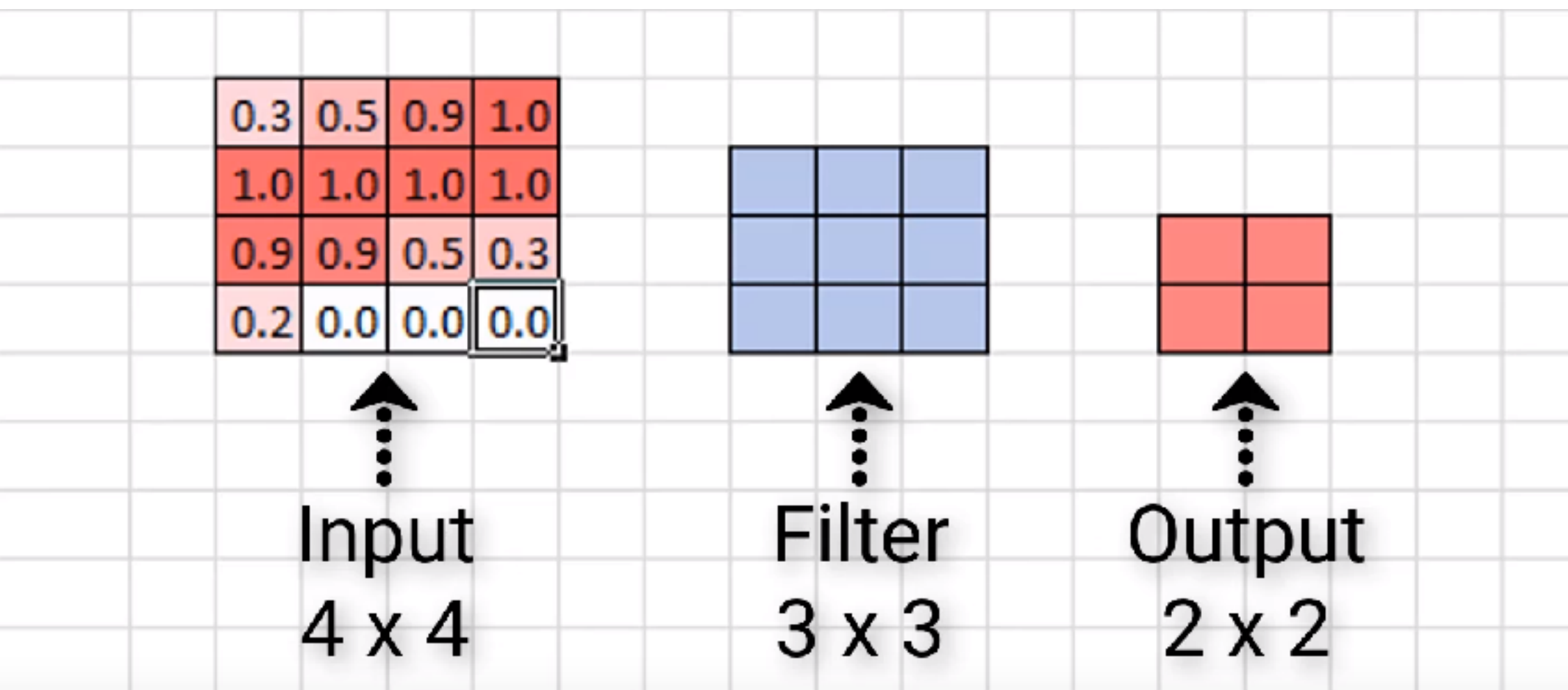


<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>

Padding

Valid (tanpa padding)

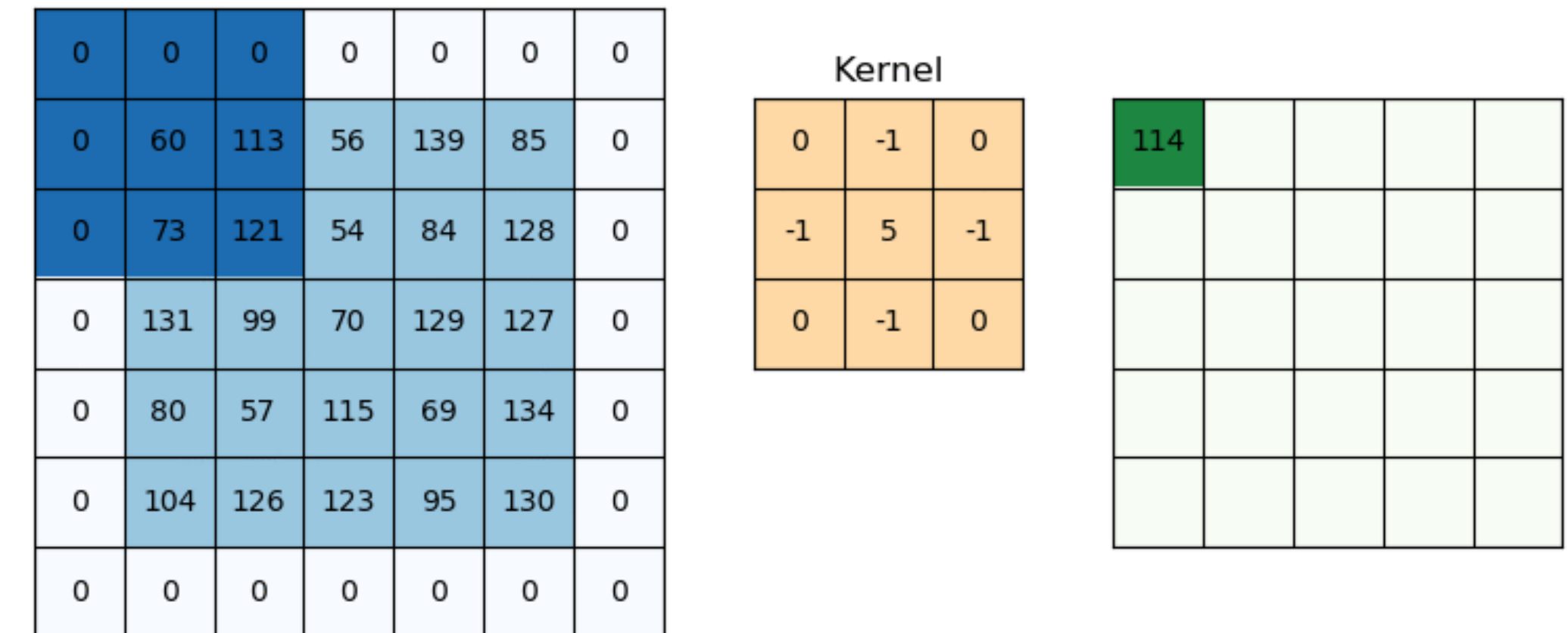
Akan menghasilkan output dengan size lebih kecil



https://deeplizard.com/learn/video/qSTv_m-KFk0

Same (menggunakan padding)

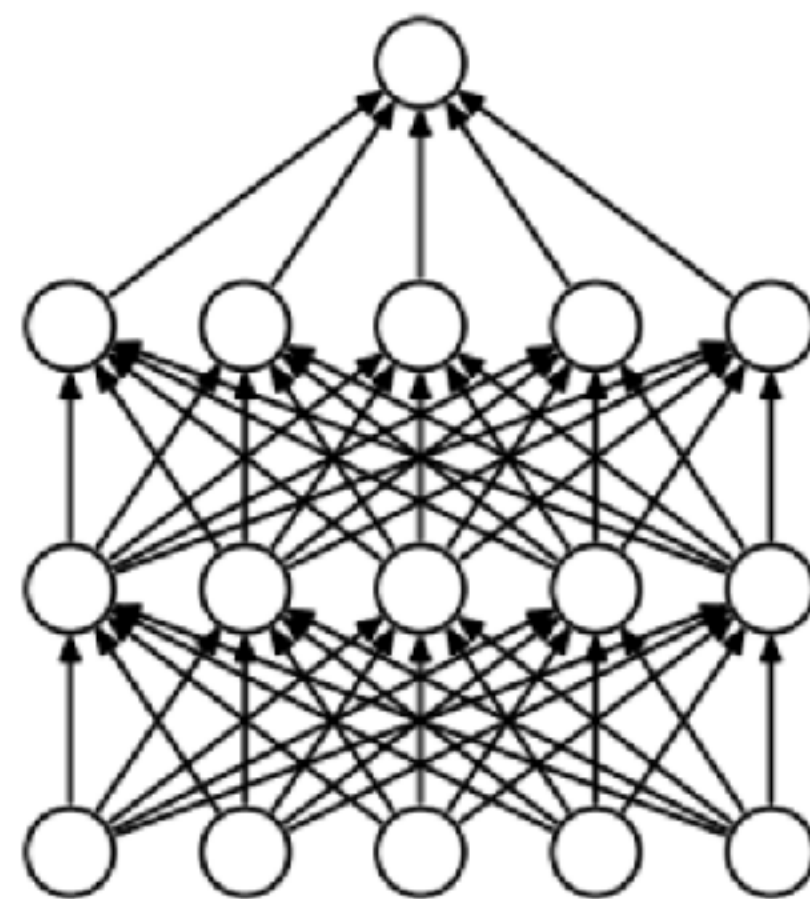
Akan menghasilkan output dengan size yang sama dengan original



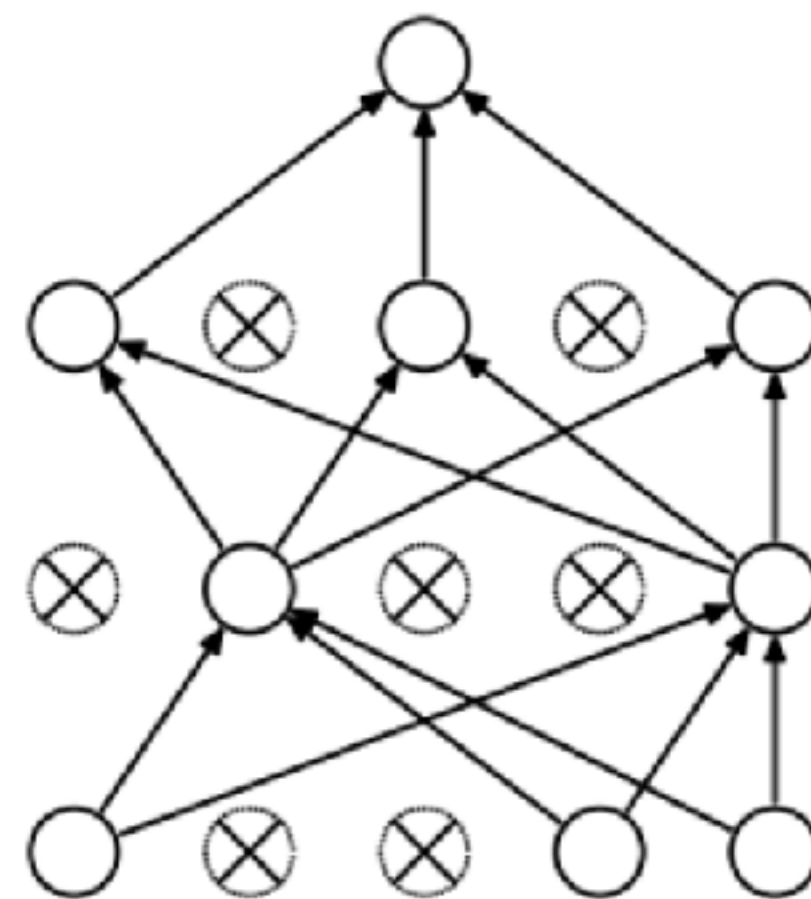
<https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/>

Regularization

- Metode untuk mencegah overfitting
- **Dropout**
 - Sengaja “mematikan” beberapa neuron
 - Digunakan untuk arsitektur dengan layer yang dalam karena rentan overfit



(a) Standard Neural Net



(b) After applying dropout.

```
tf.keras.layers.Dropout(rate)
```

Dimana *rate* adalah nilai desimal, berapa persen neuron yang akan sengaja dimatikan.

Nilai yang biasa dipakai: **0.5** atau **0.25**

<https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>

Regularization

- **Batch Normalization**

- Melakukan normalisasi (menggunakan mean & standard deviation) terhadap input untuk setiap batch, sehingga nilai minimum pada loss function dapat lebih mudah diketahui
 - Disisipkan diantara 2 layer
 - Memastikan perhitungan di setiap neuron dalam range yang sama
 - Karena data dinormalisasi, terdapat perubahan pada data input (noise tambahan) yang memberikan efek regularization

`tf.keras.layers.BatchNormalization()`

<https://kharshit.github.io/blog/2018/12/28/why-batch-normalization>

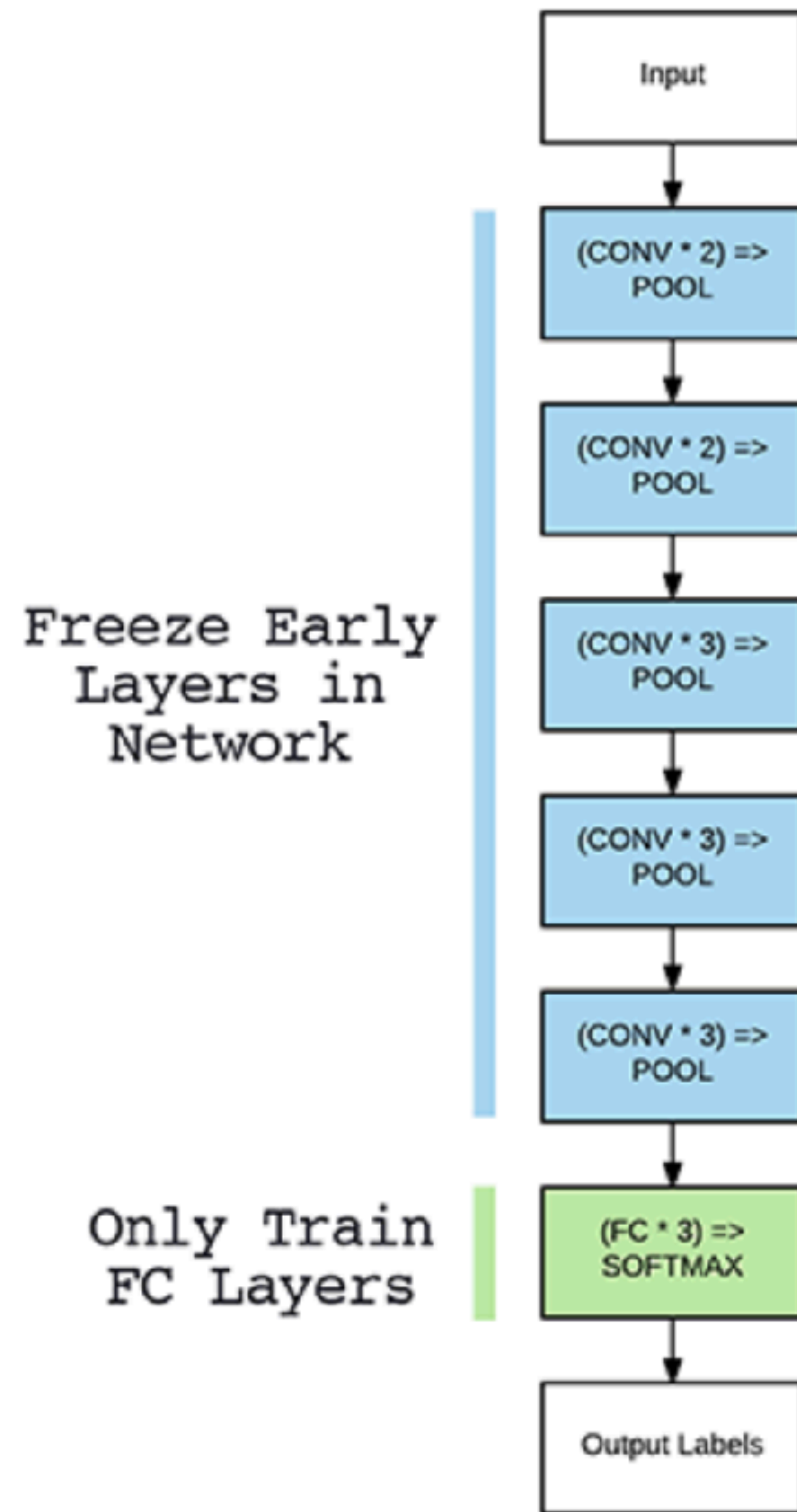
Regularization

```
# stack two more CONV layers, keeping the size of each filter
# as 3x3 but increasing to 64 total learned filters
model.add(Conv2D(64, (3, 3), padding="same",
                 kernel_initializer=init, kernel_regularizer=reg))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), strides=(2, 2), padding="same",
                 kernel_initializer=init, kernel_regularizer=reg))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Dropout(0.25))

# increase the number of filters again, this time to 128
model.add(Conv2D(128, (3, 3), padding="same",
                 kernel_initializer=init, kernel_regularizer=reg))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), strides=(2, 2), padding="same",
                 kernel_initializer=init, kernel_regularizer=reg))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Dropout(0.25))
```

<https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/>

Fine-Tuning



- Melakukan training hanya pada Fully Connected layer
- Untuk convolutional layer, semua parameter yang sudah dipelajari dibiarkan
- Apabila model sudah pernah ditrain untuk mengenal sekumpulan objek, kita dapat menggunakan **insight** yang didapat untuk mengenal objek lainnya

<https://www.pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/>

Fine-Tuning

```
from tensorflow.keras.applications import vgg16
from tensorflow.keras.layers import Dropout

vgg_conv = vgg16.VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

for layer in vgg_conv.layers[:]:
    layer.trainable = False

fine_tuned_model = Sequential()


fine_tuned_model.add(vgg_conv)

fine_tuned_model.add(Flatten())
fine_tuned_model.add(Dense(1024, activation='relu'))
fine_tuned_model.add(Dropout(0.5))
fine_tuned_model.add(Dense(1, activation='sigmoid'))

fine_tuned_model.summary()

opt = SGD(lr=0.001, momentum=0.9)

fine_tuned_model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```



Fully Connected Layer