

Introduction to Computer Vision: Neural Networks, Image Classification, Semantic Segmentation

Navasardyan Shant



November 7, 2019

- 1 Image Classification Network Architectures
- 2 Semantic Segmentation

1 Image Classification Network Architectures

2 Semantic Segmentation

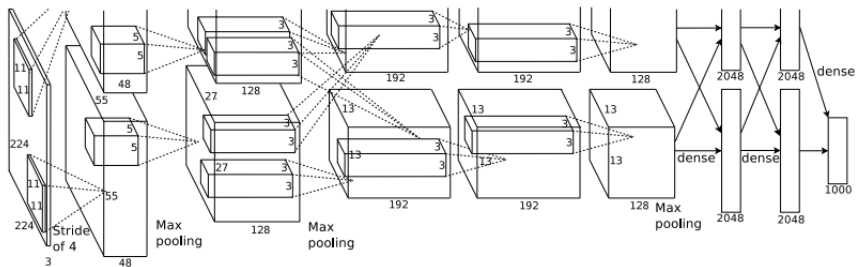


Figure: In the AlexNet architecture¹ after each convolution we have the **ReLU** activation, and after each of the first two convolutions we have **ReLU** activation, **local response normalization**. All Max-Pooling layers have the kernel size 3×3 and are done with strides 2×2 . After each fully connected layer we have the **ReLU** activation and **dropout**.

¹Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton - Advances in neural information processing systems, 2012

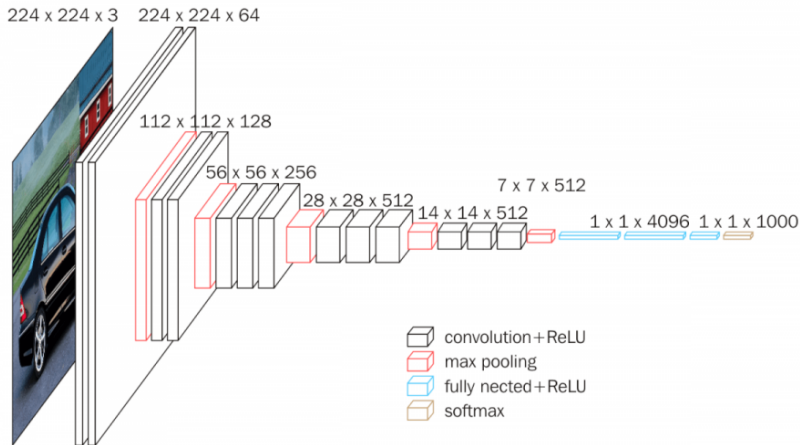


Figure: Vgg-16²

²Very deep convolutional networks for large-scale image recognition K Simonyan, A Zisserman - arXiv preprint arXiv:1409.1556, 2014

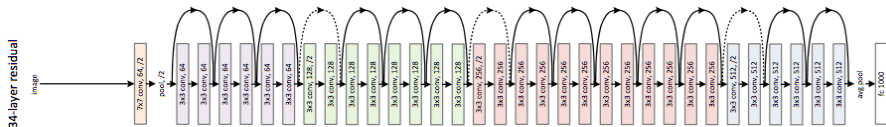


Figure: The ResNet architecture³, dotted lines means decreasing-size residual blocks. There are two kinds of residual blocks: with and without 2-strided convolution block in the residual branch

³He, Kaiming et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(2015): 770-778.

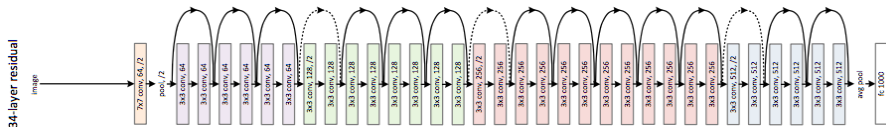


Figure: The ResNet architecture³, dotted lines means decreasing-size residual blocks. There are two kinds of residual blocks: with and without 2-strided convolution block in the residual branch

Question

How to pass the output of a convolutional layer as the input of a dense layer? What about the cases when we want our neural network to deal with images of arbitrary size?

³He, Kaiming et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 770-778.

Inception v1: GoogleNet

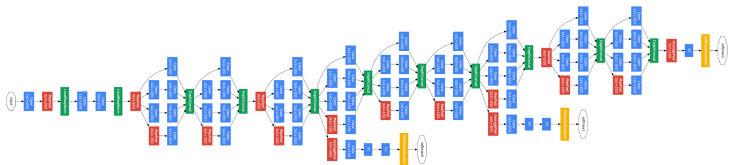


Figure: The GoogleNet architecture⁴ and the Inception Module (below) used in GoogleNet

⁴Szegedy, Christian et al. "Going deeper with convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014): 1-9.

Inception v1: GoogleNet

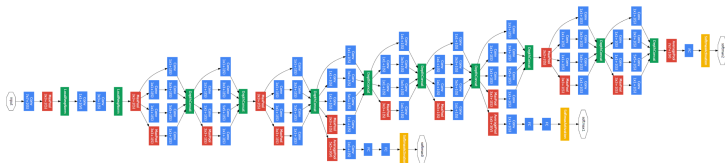
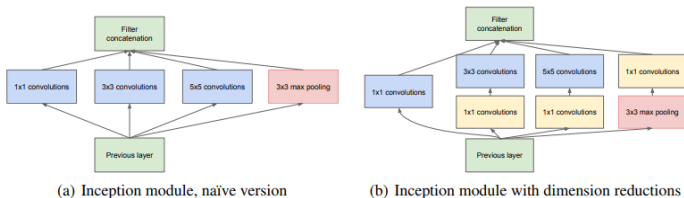


Figure: The GoogleNet architecture⁴ and the Inception Module (below) used in GoogleNet



⁴Szegedy, Christian et al. "Going deeper with convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014): 1-9.

Inception v2 and v3

The Inception v2⁵ architecture simply is a slightly modified version of GoogleNet with **batch normalization** layers before each activation layer. In the Inception v3⁶ architecture we are getting familiar with the idea of **factorizing** convolutions.

⁵Ioffe, Sergey and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv abs/1502.03167 (2015): n. pag.

⁶Szegedy, Christian et al. "Rethinking the Inception Architecture for Computer Vision." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 2818-2826.

Inception v2 and v3

The Inception v2⁵ architecture simply is a slightly modified version of GoogleNet with **batch normalization** layers before each activation layer. In the Inception v3⁶ architecture we are getting familiar with the idea of **factorizing** convolutions. Before talking about the concept of convolution factorization, let's discuss a fundamental concept in deep learning, namely **receptive field of a network**.

⁵Ioffe, Sergey and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv abs/1502.03167 (2015): n. pag.

⁶Szegedy, Christian et al. "Rethinking the Inception Architecture for Computer Vision." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 2818-2826.

Receptive Field

Let we have a convolutional neural network with consequent layers F_0, F_1, \dots, F_D (F_0, F_D are inputs and outputs of the network respectively).

Receptive Field

The **Receptive Field** of a layer F_k *with respect to* the layer F_m ($m < k$) is the maximal region in the layer F_m each element of which is contributed in forming *one* of pixels in F_D , **considering only convolutional layers as contribution**.

Receptive Field

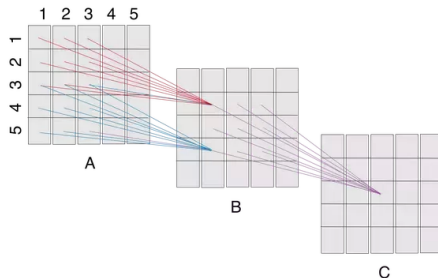
Let we have a convolutional neural network with consequent layers F_0, F_1, \dots, F_D (F_0, F_D are inputs and outputs of the network respectively).

Receptive Field

The **Receptive Field** of a layer F_k *with respect to* the layer F_m ($m < k$) is the maximal region in the layer F_m each element of which is contributed in forming *one* of pixels in F_D , **considering only convolutional layers as contribution**. By **receptive field** of a network we mean the receptive field of its output with respect to its input.

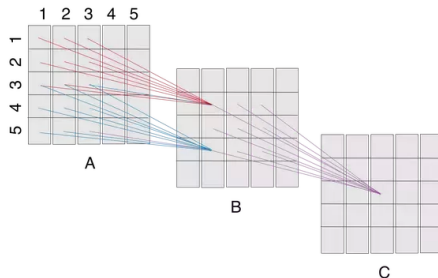
Receptive Field

Here is the illustration of the receptive field of the layer *C* with respect to the layer *A*. Here you can see that the whole layer *A* is this receptive field.



Receptive Field

Here is the illustration of the receptive field of the layer C with respect to the layer A. Here you can see that the whole layer A is this receptive field.



Note

Although the receptive field can be referred as the *"area under vision of the net"*, not all pixels in receptive field are *"equally contributed"* in formation of an output pixel. So there is a concept of **effective receptive field**^a.

^aYu, Fisher and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions." CoRR abs/1511.07122 (2015): n. pag.

Inception-V3 Revisited

So the natural need arises to enlarge the size of the receptive field of the network, or keep the size of receptive field the same but decrease the number of computations. For the latter purpose the approach of *convolution factorization* is appropriate.

Inception-V3 Revisited

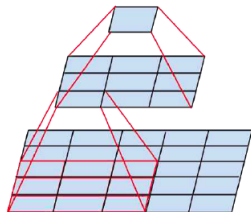
So the natural need arises to enlarge the size of the receptive field of the network, or keep the size of receptive field the same but decrease the number of computations. For the latter purpose the approach of *convolution factorization* is appropriate.

The concept of convolution factorization was introduced in the *Inception-V3* architecture. The idea is the following: in some places

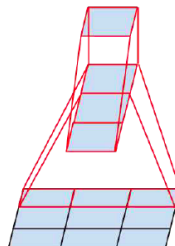
- replace 5×5 convolution layers with two 3×3 convolution layers
- replace $n \times n$ convolution layers with asymmetric consequent convolution layers of sizes $n \times 1$ and $1 \times n$.

Below you can see visualizations of these convolution factorizations.

Inception-V3: Convolution Factorization

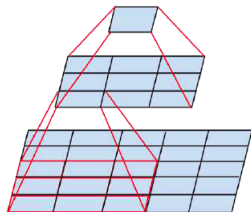


Two 3×3 convolutions replacing one 5×5 convolution

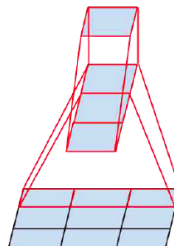


One 3×1 convolution followed by one 1×3 convolution replaces one 3×3 convolution

Inception-V3: Convolution Factorization



Two 3x3 convolutions replacing one 5x5 convolution



One 3x1 convolution followed by one 1x3 convolution replaces one 3x3 convolution

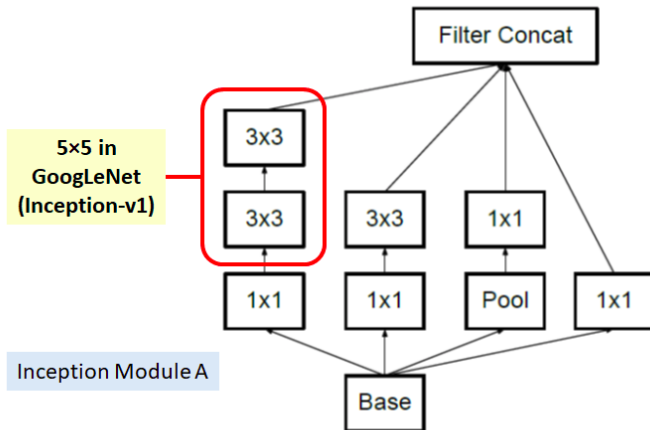
You can see that the receptive fields of replaced blocks are the same as before replacement.

Inception-V3: Convolution Factorization

So the following types of inception modules are introduced in the architecture of Inception-V3.

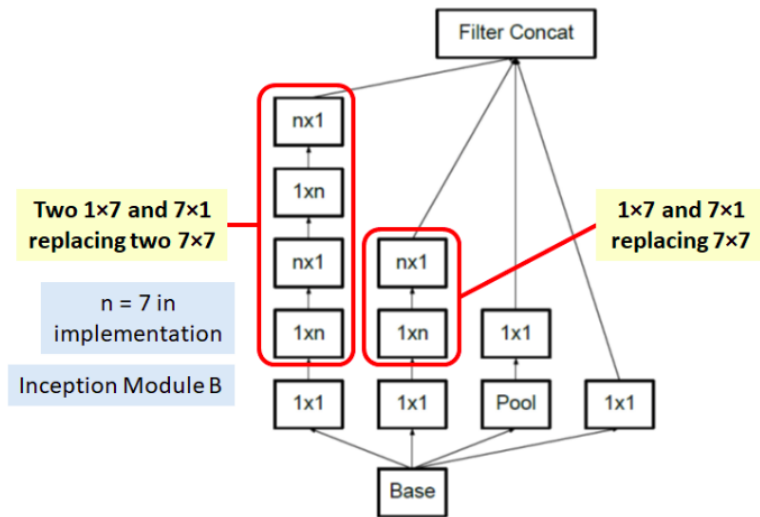
Inception-V3: Convolution Factorization

So the following types of inception modules are introduced in the architecture of Inception-V3.



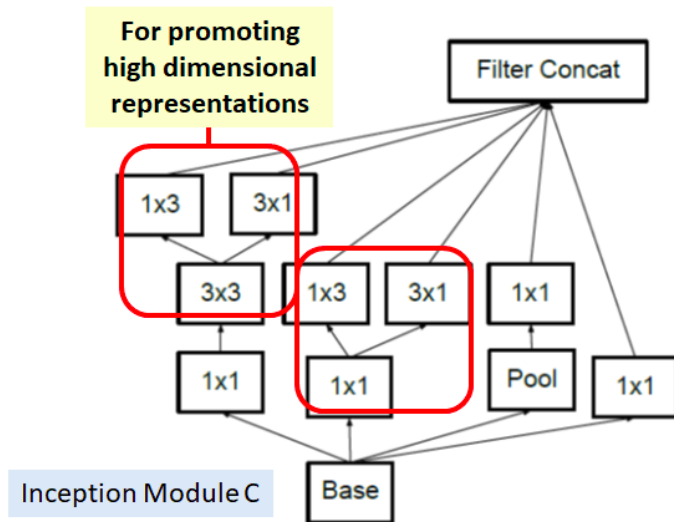
Inception Module A using factorization

Inception-V3: Convolution Factorization



Inception Module B using asymmetric factorization

Inception-V3: Convolution Factorization



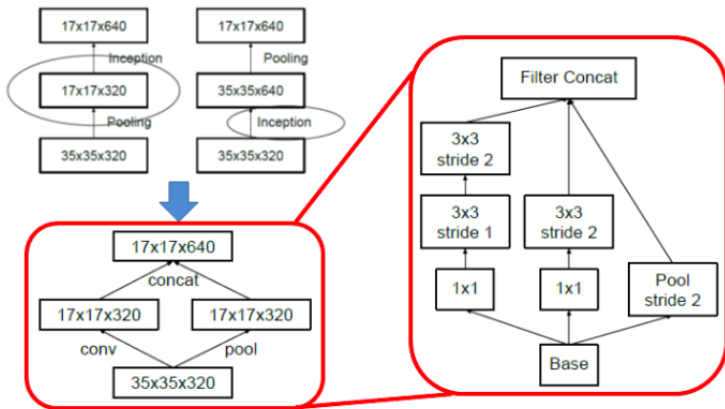
Inception Module C using asymmetric factorization

Inception-V3: Grid Size Reduction

The architecture of Inception-V3 also introduces a new size reduction method, directly in the inception block.

Inception-V3: Grid Size Reduction

The architecture of Inception-V3 also introduces a new size reduction method, directly in the inception block.



Conventional downsizing (Top Left), Efficient Grid Size Reduction (Bottom Left), Detailed Architecture of Efficient Grid Size Reduction (Right)

Inception-V3

So the whole architecture of the network Inception-V3 is the following:

Inception-V3

So the whole architecture of the network Inception-V3 is the following:

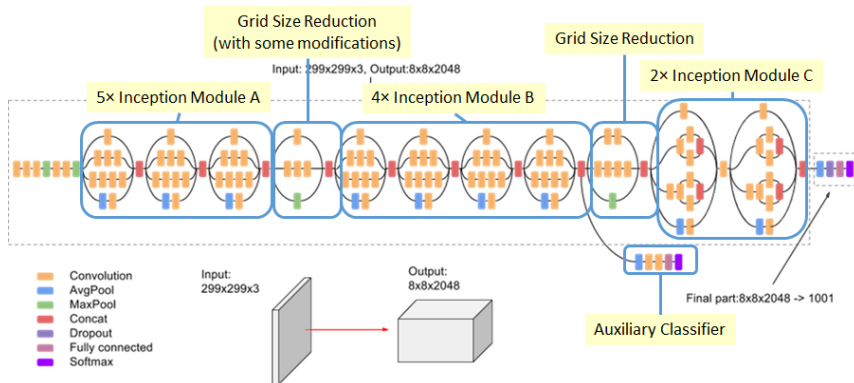


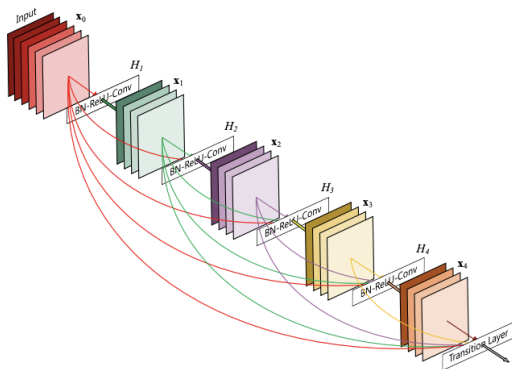
Figure: After each convolution layer we use BatchNorm and ReLU

Another architecture similar to ResNet architecture is introduced as *DenseNet*⁷. Here you can see the main block of this network, called *dense block*.

⁷Huang, Gao et al. "Densely Connected Convolutional Networks." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 2261-2269.

DenseNet

Another architecture similar to ResNet architecture is introduced as *DenseNet*⁷. Here you can see the main block of this network, called *dense block*.

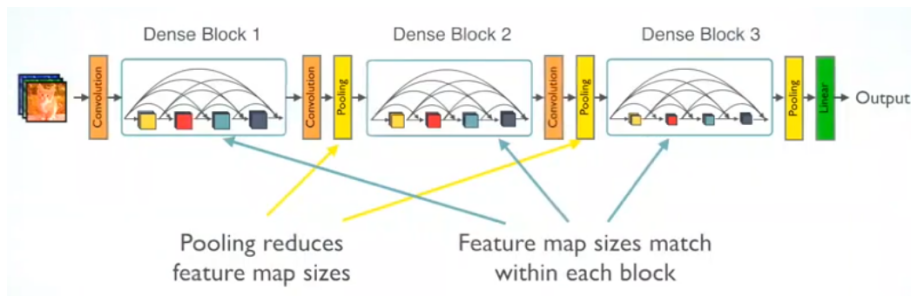


⁷Huang, Gao et al. "Densely Connected Convolutional Networks." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 2261-2269.

The subsampling in this network is done in so called *transition* layer, which is convolution, followed by average pooling.

DenseNet

The subsampling in this network is done in so called *transition* layer, which is convolution, followed by average pooling.



The model SqueezeNet⁸ aims to reduce the model size and computation complexity while preserving the high accuracy. As described in the paper, there are several strategies to achieve a high accuracy with a small model:

⁸Iandola, Forrest N. et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size." ArXiv abs/1602.07360 (2017).

The model SqueezeNet⁸ aims to reduce the model size and computation complexity while preserving the high accuracy. As described in the paper, there are several strategies to achieve a high accuracy with a small model:

- use more 1×1 convolutions instead of 3×3 convolutions

⁸Iandola, Forrest N. et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size." ArXiv abs/1602.07360 (2017).

The model SqueezeNet⁸ aims to reduce the model size and computation complexity while preserving the high accuracy. As described in the paper, there are several strategies to achieve a high accuracy with a small model:

- use more 1×1 convolutions instead of 3×3 convolutions
- in case of 3×3 convolutions, decrease the number of its input channels

⁸Iandola, Forrest N. et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size." ArXiv abs/1602.07360 (2017).

The model SqueezeNet⁸ aims to reduce the model size and computation complexity while preserving the high accuracy. As described in the paper, there are several strategies to achieve a high accuracy with a small model:

- use more 1×1 convolutions instead of 3×3 convolutions
- in case of 3×3 convolutions, decrease the number of its input channels
- downsample in deeper features instead of earlier features.

⁸Iandola, Forrest N. et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size." ArXiv abs/1602.07360 (2017).

SqueezeNet

So the SqueezeNet architecture introduces the module called **fire module**:

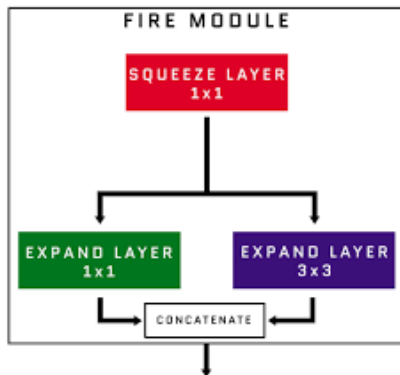
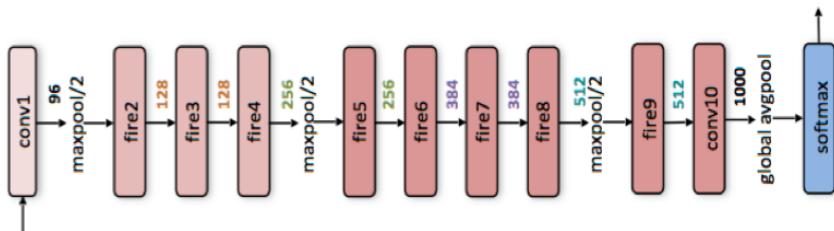


Figure: This *fire module* has three parameters. $s_{1 \times 1}$ is the number of the filters of the 1×1 convolution in the "*Squeeze Layer*", $e_{1 \times 1}$ and $e_{3 \times 3}$ are the numbers of the filters of the 1×1 and 3×3 convolutions in the "*Expand Layers*". Each layer consists of a convolution followed by *ReLU* activation.

SqueezeNet

So the SqueezeNet architecture is the following:



Another small-sized network architecture is described in *MobileNet*⁹. This is a bunch of neural networks similar to each other, so we refer to all of them as *MobileNet*.

⁹Howard, Andrew G. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." ArXiv abs/1704.04861 (2017): n. pag.

Another small-sized network architecture is described in *MobileNet*⁹. This is a bunch of neural networks similar to each other, so we refer to all of them as *MobileNet*.

Depthwise Separable Convolutions

In the *MobileNet* architecture the core idea is the concept of **Depthwise Separable Convolution** block. This block consists of two convolutional blocks: **depthwise** block and **pointwise** block.

⁹Howard, Andrew G. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." ArXiv abs/1704.04861 (2017): n. pag.

Another small-sized network architecture is described in *MobileNet*⁹. This is a bunch of neural networks similar to each other, so we refer to all of them as *MobileNet*.

Depthwise Separable Convolutions

In the *MobileNet* architecture the core idea is the concept of **Depthwise Separable Convolution** block. This block consists of two convolutional blocks: **depthwise** block and **pointwise** block. The **depthwise** convolution applies a *single* filter to each input channel, so the output of depthwise convolution has the same number of channels as the input. The **depthwise** block consists of a depthwise convolution followed by *Batch Normalization* and *ReLU* activation.

⁹Howard, Andrew G. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." ArXiv abs/1704.04861 (2017): n. pag.

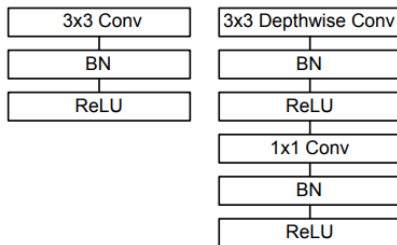
Another small-sized network architecture is described in *MobileNet*⁹. This is a bunch of neural networks similar to each other, so we refer to all of them as *MobileNet*.

Depthwise Separable Convolutions

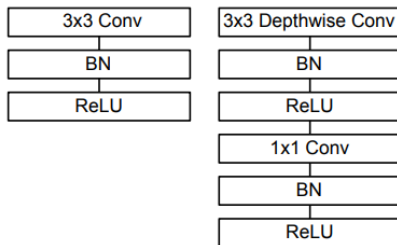
In the *MobileNet* architecture the core idea is the concept of **Depthwise Separable Convolution** block. This block consists of two convolutional blocks: **depthwise** block and **pointwise** block. The **depthwise** convolution applies a *single* filter to each input channel, so the output of depthwise convolution has the same number of channels as the input. The **depthwise** block consists of a depthwise convolution followed by *Batch Normalization* and *ReLU* activation. The **pointwise** convolution is just a 1×1 convolution. The **pointwise** block consists of a pointwise convolution followed by *Batch Normalization* and *ReLU* activation.

⁹Howard, Andrew G. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." ArXiv abs/1704.04861 (2017): n. pag.

Left: standard convolutional block. Right: depthwise separable convolutional block



Left: standard convolutional block. Right: depthwise separable convolutional block



Width Multiplier

The number of filters of the pointwise 1×1 convolutions can be controlled with a hyperparameter called **width multiplier**, which is denoted by α . If in the baseline *MobileNet* architecture we have 1×1 convolution layers with numbers of filters C_1, C_2, C_3, \dots , then in the *MobileNet* - α architecture we have these numbers multiplied by α , i.e. the the numbers of filters of 1×1 convolutions are $\alpha C_1, \alpha C_2, \alpha C_3, \dots$

1 Image Classification Network Architectures

2 Semantic Segmentation

Semantic Segmentation

Semantic Segmentation

As previously, we refer to the image segmentation problem as a partitioning the image into regions (not necessarily connected).

Semantic Segmentation

As previously, we refer to the image segmentation problem as a partitioning the image into regions (not necessarily connected). The task of **semantic image segmentation** is in addition of partitioning the image also label the formed regions. The word *semantic* means that we desire to label these regions *semantically*, i.e. in such a way, that we, humans, understand objects we see.

Semantic Segmentation

As previously, we refer to the image segmentation problem as a partitioning the image into regions (not necessarily connected). The task of **semantic image segmentation** is in addition of partitioning the image also label the formed regions. The word *semantic* means that we desire to label these regions *semantically*, i.e. in such a way, that we, humans, understand objects we see. So, as the task of semantic segmentation is to get some semantic partition of the image and label the formed regions, we can combine these two steps into **classification of each pixel in the image semantically**.

Semantic Segmentation

Semantic Segmentation

As previously, we refer to the image segmentation problem as a partitioning the image into regions (not necessarily connected). The task of **semantic image segmentation** is in addition of partitioning the image also label the formed regions. The word *semantic* means that we desire to label these regions *semantically*, i.e. in such a way, that we, humans, understand objects we see. So, as the task of semantic segmentation is to get some semantic partition of the image and label the formed regions, we can combine these two steps into **classification of each pixel in the image semantically**.

Therefore, **semantic image segmentation in essence is a classification of each pixel in the image**.

Semantic Segmantation

Note

As the semantic segmentation task is a classification task, the main objective we want to minimize remains **categorical cross-entropy** loss, averaged among all pixels.

Note

As the semantic segmentation task is a classification task, the main objective we want to minimize remains **categorical cross-entropy** loss, averaged among all pixels. I.e. if we have an image $I \in \mathbb{R}^{H \times W \times 3}$ and a prediction $P \in \mathbb{R}^{H \times W \times K}$ (here K is the number of classes), then the loss is computed as

$$L(I, P) = -\frac{1}{H \cdot W} \sum_{i,j} \sum_{k=1}^K GT_{i,j,k} \log(P_{i,j,k}),$$

where $GT \in \{0, 1\}^{H \times W \times K}$ is one-hot encoded tensor of *ground truth* pixel classes of the image I .

Semantic Segmentation

Note

As the semantic segmentation task is a classification task, the main objective we want to minimize remains **categorical cross-entropy** loss, averaged among all pixels. I.e. if we have an image $I \in \mathbb{R}^{H \times W \times 3}$ and a prediction $P \in \mathbb{R}^{H \times W \times K}$ (here K is the number of classes), then the loss is computed as

$$L(I, P) = -\frac{1}{H \cdot W} \sum_{i,j} \sum_{k=1}^K GT_{i,j,k} \log(P_{i,j,k}),$$


where $GT \in \{0, 1\}^{H \times W \times K}$ is one-hot encoded tensor of *ground truth* pixel classes of the image I .

Note

There are also other losses for semantic segmentation about which we will discuss later in this course.

The Fully Convolutional Network

We start the investigation of semantic segmentation algorithms with **FCNs - Fully Convolutional Networks**¹⁰

¹⁰Shelhamer, Evan et al. "Fully Convolutional Networks for Semantic Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2014): 640-651. 


The Fully Convolutional Network

We start the investigation of semantic segmentation algorithms with **FCNs - Fully Convolutional Networks**¹⁰

FCN

Fully Convolutional Networks can take inputs of **arbitrary** size.

At first we adapt a classifier network to the fully convolutional one. For this we can refer to each dense layer as a convolution layer with kernel size as the size of the preceding layer. Or we can just throw away the part after the last convolutional layer.

¹⁰Shelhamer, Evan et al. "Fully Convolutional Networks for Semantic Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2014): 640-651. 


The Fully Convolutional Network

We start the investigation of semantic segmentation algorithms with **FCNs - Fully Convolutional Networks**¹⁰

FCN

Fully Convolutional Networks can take inputs of **arbitrary** size.

At first we adapt a classifier network to the fully convolutional one. For this we can refer to each dense layer as a convolution layer with kernel size as the size of the preceding layer. Or we can just throw away the part after the last convolutional layer. In both cases the question arises how to **upsample** the resulting tensor to the input image size. This can be done, for example, with bilinear upsampling. Also a method called **deconvolution or transposed convolution** is used. The latter enables us also to train these upsampling parts of the network. We will talk about transposed convolutions later.

¹⁰Shelhamer, Evan et al. "Fully Convolutional Networks for Semantic Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2014): 640-651. 

The Fully Convolutional Network

As just upsampling the last convolution layer's output gives unsatisfying coarse results, we can make so called **skip-connections** from the "*finer*" layers to the "*coarser*" layers. These skip-connections are illustrated in the image below

The Fully Convolutional Network

As just upsampling the last convolution layer's output gives unsatisfying coarse results, we can make so called **skip-connections** from the "finer" layers to the "coarser" layers. These skip-connections are illustrated in the image below

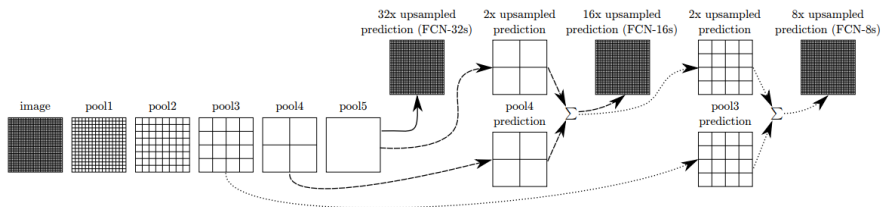


Figure: Pools are the pooling layers of some fully convolutional network, adapted from a classification network

Deconvolutions or Transposed Convolutions

Dilated Convolutions

Introduction to Computer Vision: Neural Networks, Image Classification, Semantic Segmentation

Navasardyan Shant



November 7, 2019