DenseNets

Pablo Ruiz – Harvard University – August 2018

Background

Densely Connected Convolutional Networks [1], DenseNets, are the next step on our way to expand the depth of deep convolutional networks.

We have seen how we have gone from LeNet with 5 layers, to VGG with 19 layers and ResNets surpassing 100 even 100 layers.

Motivation

The problems arise with CNNs when they go deeper. This is because the ***path*** for information from the input layer until the output layer (and for the gradient in the opposite direction) ***becomes so big***, that they can ***get vanished*** before reaching the other side.

DenseNets **simplify the connectivity pattern** between layers introduced in other architectures:

- Highway Networks [2]
- Residual Networks [3]
- Fractal Networks [4]

The authors solve the problem ensuring **maximum information (and gradient) flow**. To do it, they simply connect every layer directly with each other.

Instead of drawing representational power from extremely deep or wide architectures, DenseNets exploit the potential of the network through feature reuse

---

What problem DenseNets solve?

Counter-intuitively, by connecting this way DenseNets ***require fewer parameters*** than an equivalent traditional CNN, as there is ***no need to learn redundant feature maps***.

Furthermore, some variations of ResNets have proven that many layers are barely contributing and can be dropped. In fact, the number of parameters of ResNets are big because every layer has its weights to learn. Instead, ***DenseNets layers are very narrow*** (e.g. 12 filters), and they just ***add a small set of new feature-maps***.

Another problem with very deep networks was the problems to train, because of the mentioned flow of information and gradients. DenseNets solve this issue since ***each layer has direct access to the gradients from the loss function*** and the original input image.

Structure

Traditional feed-forward neural networks connect the output of the l^{th} layer to the next $(l+1)^{th}$ layer after applying a ***composite of operations*** $H_l(\cdot)$. We have already seen that normally this composite includes a convolution operation or pooling layers, a batch normalization and an activation function. The equation for this would be:

$$x_l = H_l(x_{l-1})$$

ResNets extended this behavior including the skip connection, reformulating this equation into:

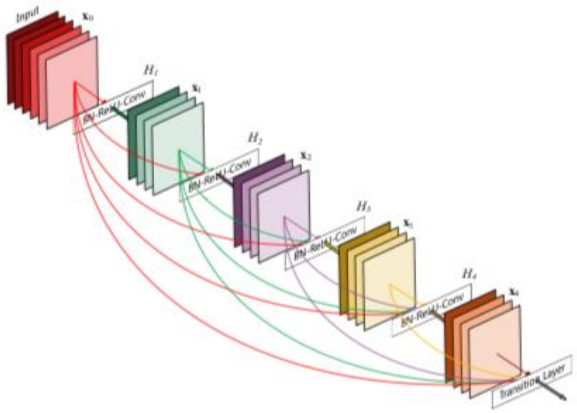

Figure 1. DenseNet with 5 layers with expansion of 4. [1]

$$x_l = H_l(x_{l-1}) + x_{l-1}$$

DenseNets make the first difference with ResNets right here. **DenseNets do not sum the output feature maps of the layer with the incoming feature maps but concatenate them**.

Consequently, the equation reshapes again into:

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

The same problem we faced on our work on ResNets, this grouping of feature maps cannot be done when the sizes of them are different. Regardless if the grouping is an addition or a concatenation. Therefore, and the same way we used for ResNets, DenseNets are divided into **DenseBlocks, where the dimensions of the feature maps remains constant within a block, but the number of filters and changes between them**. These layers between them are called ***Transition Layers*** and take care of the downsampling applying a batch normalization, a 1x1 convolution and a 2x2 pooling layers.

Now we are ready to talk about the **growth rate.** Since we are concatenating feature maps, this channel dimension is increasing at every layer. If we make H_l to produce k feature maps every time, then we can generalize for the l^{th} layer:

$$k_l = k_0 + k * (l-1)$$

This hyperparameter k is the growth rate. The growth rate regulates how much information is added to the network each layer. How so?

We could see the feature maps as the information of the network. ***Every layer has access to its preceding feature maps***, and therefore, to the ***collective knowledge***. Each layer is then adding a new information to this collective knowledge, in concrete k feature maps of information.

DenseNets-B

DenseNets-B are just regular DenseNets that take advantage of 1x1 convolution to reduce the feature maps size before the 3x3 convolution and improve computing efficiency. The B comes after the name Bottleneck layer you are already familiar with from the work on ResNets.

DenseNets-BC

DenseNets-C are another little incremental step to DenseNets-B, for the cases where we would like to ***reduce the number of output feature maps***. The ***compression factor*** θ determines this reduction. Instead of having m feature maps at a certain layer, we will have θm. Of course, θ is in the range $[0 - 1]$. So DenseNets will remain the same when $\theta = 1$, and will be DenseNets-B otherwise.

To be aligned with the work on ResNets, we will go into the details for the case on ImageNet dataset. However, the structure is simpler for CIFAR-10 or SVHN since the input volumes are smaller.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 2. Sizes of outputs and convolutional kernels for different DenseNets [1] architectures on ImageNet.

As you know from previous works on other architectures, I prefer to observe how actually the volumes that are going through the model are changing their sizes. This way is easier to understand the mechanism of a particular model, to be able to adjust it to our particular needs. I will try to follow the notation close to the PyTorch official implementation to make it easier to later implement it on PyTorch.

However, because of the highly dense number of connections on the DenseNets, the visualization gets a little bit more complex that it was for VGG and ResNets. **Error! Reference source not found.** shows a very simple scheme on the architecture of the DenseNet-121, which will be the DenseNet we will focus on over this work. This is because it is the simples DenseNet among those designed over the ImageNet dataset.

We can compare the **Error! Reference source not found.** with the Figure 2 on DenseNet-121. The measures under each volume represent the sizes of the width and depth, whereas the numbers on top represents the feature maps dimension. I included how they are derived to help us understand better incoming steps.

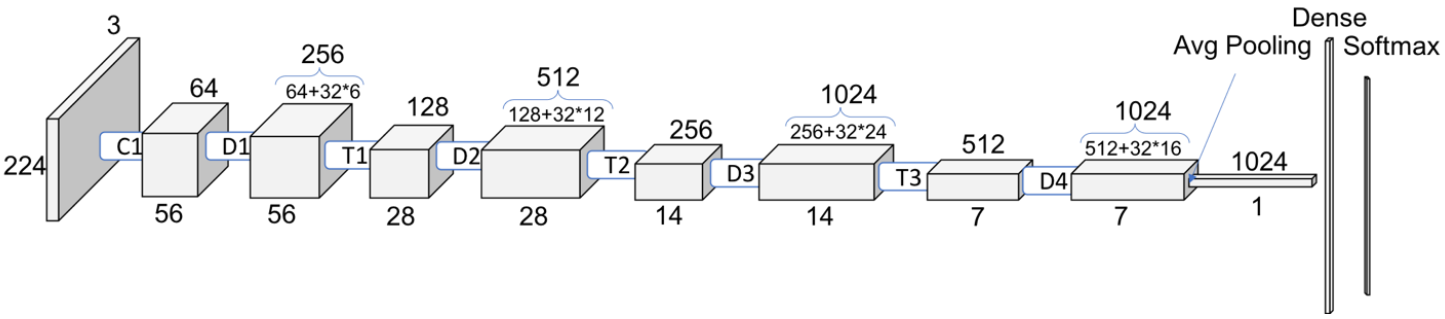

Figure 3. Another look at Dense-121. Dx: Dense Block x. Tx: Transition Block x.

Dense and Transition Blocks

You may be wondering where the bypass connections are all over the DenseNet, and you are asking right! However, I just wanted to first draw the simplest scheme possible to know go deeper on the whole structure of the DenseNet. But just for the curious, you can notice how the first number of the addition to calculate the feature maps of every new volume matches the feature maps dimension of the previous volume. This will explain the bypass connection because it precisely means that we are **concatenating** (concatenate means add dimension, but not add values!) **new information to the previous volume**, which is being **reused**.

Note that that **32 is precisely the growth rate** we mentioned at the beginning of this section. So, what can we get from the above picture?

> ❖ The volume after every Dense Block increase by the growth rate times the number of Dense Layers within that Dense Block

In Figure 4Figure 4 we go deeper now and understand what is actually happening inside every block.

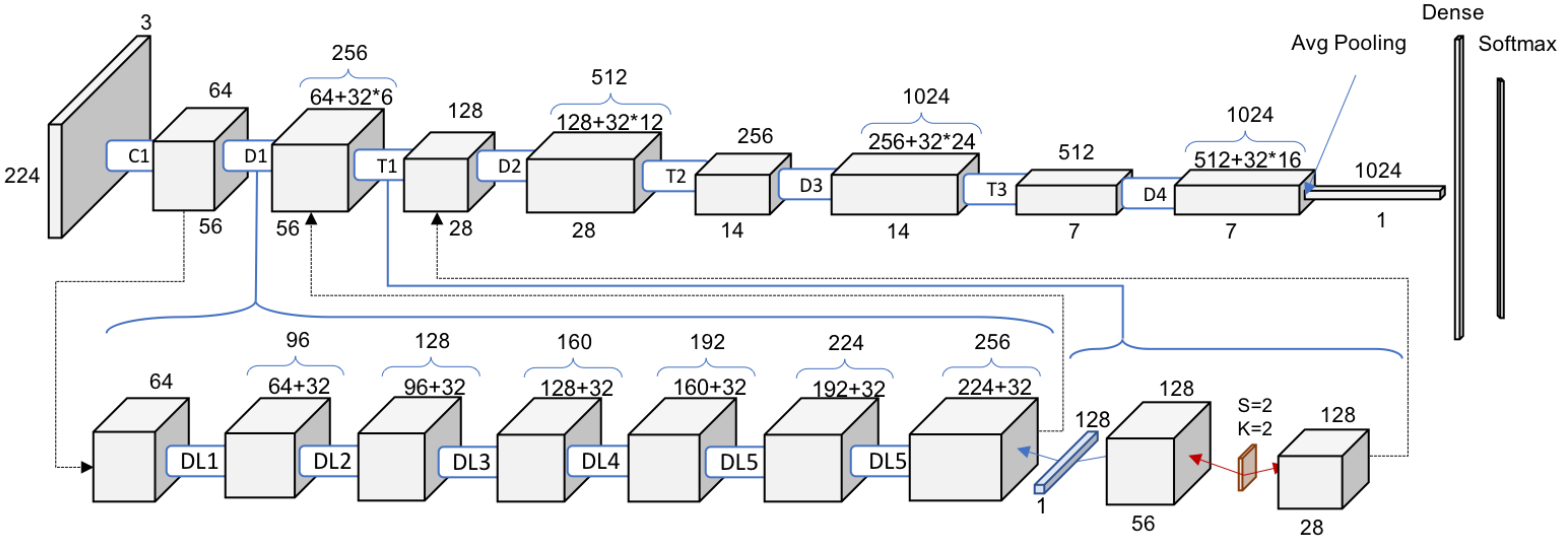

Figure 4. One level deeper look at DenseNet-121. Dense Block and Transition Block. DLx: Dense Layer x

Dense Layers

Now we can understand the statement above. Every layer is adding to the previous volume these 32 new feature maps. This is why we go from 64 to 256 after 6 layers. In addition, Transition Block performs as 1x1 convolution with 128 filters. followed by a 2x2 pooling with a stride of 2, resulting on dividing the size of the volume and the number of feature maps on half. We can make new statements from this pattern observation.

- The volume within a Dense Block remains constant
- The volume and the feature maps are halved after every Transition Block

However, we still can go 1 level deeper! We need to understand what is going on inside every Dense Layer within each Dense Block, since it is not trivial yet. Let's go then for the full picture!

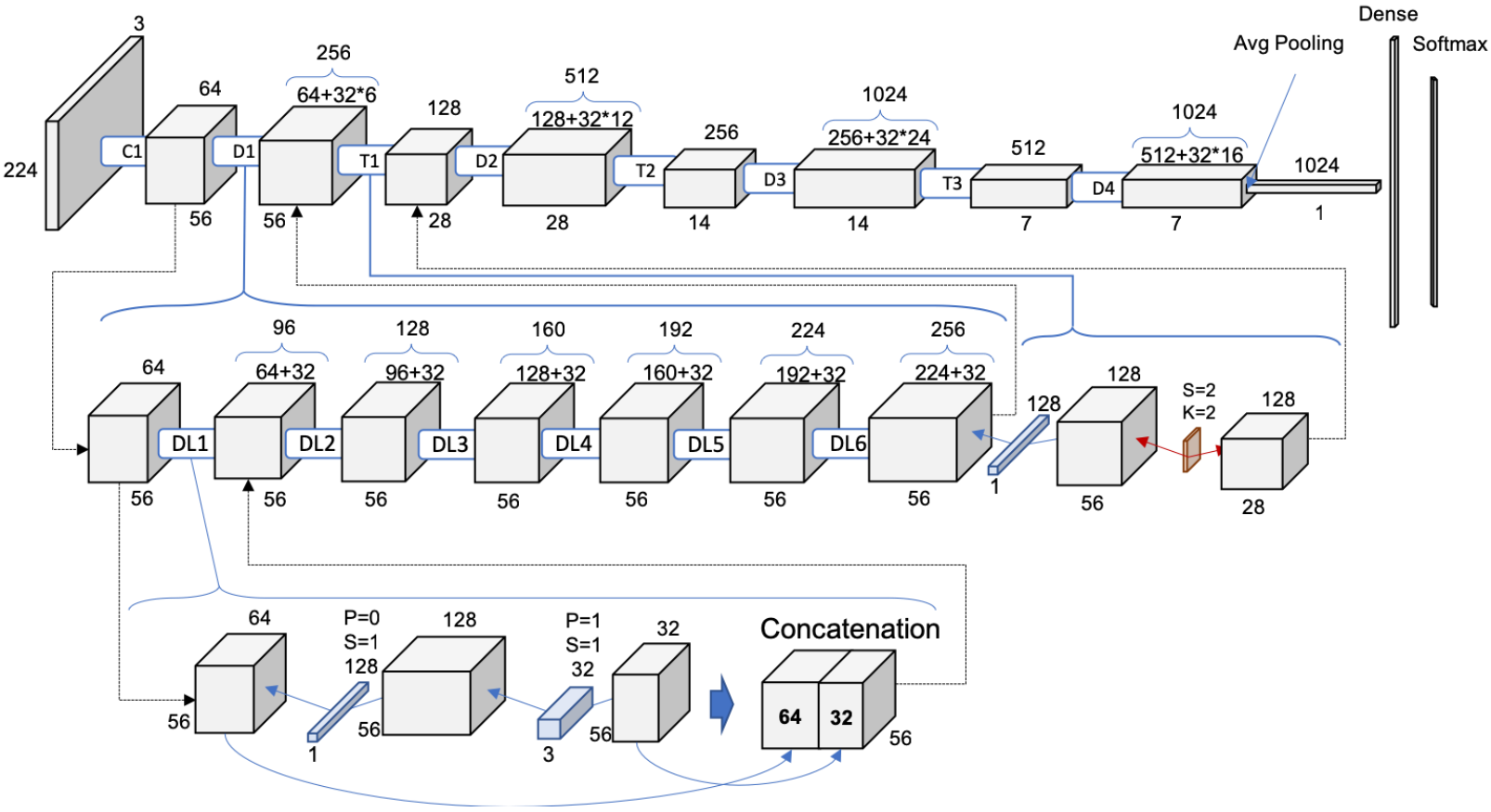

Figure 5. 2 level deep. Full schematic representation of DenseNet-121

In the new deeper level representing the first Dense Layer within the first Dense Block, we can see how actually this behavior of adding 32 times the number of layers is achieved. We perform as the authors suggest a 1x1 convolution with 128 filters to reduce* the feature maps size and the perform a more expensive 3x3 convolution (remember to include the padding to ensure the dimensions remain constant) with this chosen 32 number of feature maps of growth rate.

Then, the input volume and the result of the two operations (which are the same for every Dense Layer within every Dense Block) are concatenated, in the action of adding new information to the common knowledge of the network.

Works Cited

[1] G. Huang, Z. Liu and L. van der Maaten, "Densely Connected Convolutional Networks," 2018.

[2] R. Kumar, K. Gredd and J. Schmidhuber, "Highway Networks," 2015.

[3] K. He, X. Zhang, S. Ren and Jian Sun, "Deep Residual Learning for Image Recognition," 2015.

[4] G. Larsson, M. Maire and G. Shakhnarovich, "FractalNet: Ultra-Deep Neural Networks without Residuals," 2017.