DenseNet Discussion Summary

Presentation Summary

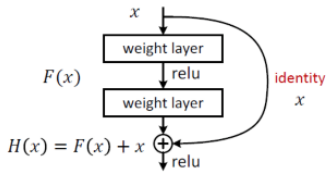
Resnet Motivation

Naively stacking more layers (making a network deeper) causes problems:

- Exploding and Vanishing gradients
    - Backpropagation compounds gradients, these can “explode” by getting too big
    - Or these can tend towards zero and effectively “vanish”.
    - Good gradients are responsible for effective learning
- Accuracy degrades with deeper networks
    - This is NOT due to overfitting as deeper networks were found to have high training error than that of shallow networks.
    - This happens because the optimization is much harder in deeper networks

Resnet Key Ideas:

- Resnet introduces skip connections in order to propagate gradient and train deeper networks
    - Solves exploding/vanishing gradient problem while not adding parameters to the network (no added complexity)
    - Makes it much easier for the solver to optimize
    - Deeper networks are not degraded anymore



- Skip connections enforces dimensions to match, they overcome this by linearly projecting the features if the dimensions do not match.

Resnet Drawbacks:

The identity function and the output are combined by summation, which may impede the information flow in the network

DenseNet Key Ideas:

- Make use of skip/shortcut connections like in ResNet
- Replace addition with channel-wise concatenation in skip connections
- Densely connected layer to all previous layers
    - Aim: Improve information flow
    - Allows for strong gradient flow during backpropagation
    - Diversifies Features (as opposed to correlated features of ResNet) as feature maps can have very different distributions. Concatenating feature maps can preserve these distributions and increase the variance of the outputs, encouraging feature reuse.
- Uses pre-activation. Each layer is composed of:

    - Batch-Normalization
    - ReLU activation
    - Convolution layer
- Introduces Bottleneck layers: 1x1 convolutions
    - Aim: reduce channel dimensionality of features in deeper layers. Called DenseNet-B.
- Every few layers, dimensionality is reduced with Conv Block + Max Pooling, this is called the transition layer.
- Growth rate 'k' is a hyperparameter:
    - Feature-maps can be viewed as the global state of the network. Each layer adds k feature-maps of its own to this state.
    - The growth rate regulates how much new information each layer contributes to the global state.
    - This is known as the "Collective Knowledge" of the network.
- Compression: Aims to make a more compact network by reducing the number of feature-maps (m) at a transition layer. Called DenseNet-C.

DenseNet Drawbacks:

DenseNet is still widely used and highly effective. Specifically, it is often used as a backbone for other (bigger) networks or for transfer learning, where the pretrained model is used for a different task and only the last layer(s) are tuned. However, it is prone to overfitting in some cases and there are a lot of interconnections, on the order of L^2. Never networks like SparseNet achieves similar or better accuracy with fewer layers/interconnects.

Discussion Summary

Using concatenation instead of addition uses more weights so why doesn't DenseNet take forever to train?

Overall, there are less weights to train in a DenseNet (0.8M weights) as compared to a ResNet (10.2M).

The number of parameters as a function of number of layers seems quadratic relation. If so we would not be able to construct very deep densenet.

The number of parameters as a function of number of layers in a Dense Block is actually linear. The growth rate is a fixed parameter, k, so the number of layers is k*l.

Why is concatenation better than addition? And how does the bottleneck layer help?

The benefit of using concatenation over addition is twofold:

1) While addition requires no extra parameters, from the standpoint of the optimization engine, it "confuses" where the gradient came from. This makes optimization harder than if the error came

from a separate channel. This is why we say that concatenation provides better gradient (and information) flow.

2) Addition also requires the tensors to be the same shape/dimensionality. Indeed, if the tensor shapes do not match, a more complicated scheme is required such as projection. When using channel-wise concatenation, you are relaxing the dimension matching constraint as now only dimensions that aren't the channels need to match. To account for more or less channels, you can use a bottleneck layer, which will only affect the total number of channels.

What about transfer learning? How does DenseNet compare to Resnet? Is DenseNet still used? Would we just split it into multiple densenet blocks, and group some blocks as feature layers network used in transfer learning?

DenseNet is considered to be a very well-trained network and as such transfer learning with DenseNet is very much possible and it is widely used.

The most basic example of transfer learning would be one where the pretrained network is mostly frozen in place and only the last layer(s) is/are finetuned or replaced. Here you can see an example where this was done in order to detect lung diseases in bronchoscopy.

DenseNet is also commonly used as a backbone architecture in tasks like segmentation where an Encoder/Decoder network is usually used. When used like this, some internal activations of the network will be used as skip connections into the decoder network. In this paper, which is the current state of the art in monocular depth estimation, they use ResNet, ResNext and DenseNet as backbones in their networks. Specifically, they show empirically that the DenseNet performs better. The way they incorporate the pretrained DenseNet is interesting too: the authors use the post denseblock activations as inputs to the decoder.