CNNs

Convolutional Layers have relatively few weights and more layers than other architectures. In practice, data scientists add layers to CNNs to solve specific problems using Transfer Learning.

Transfer Learning

The main idea of Transfer Learning consists of keeping early layers of a pre-trained network and re-train the later layers for a specific application.

Last layers in the network capture features that are more particular to the specific data you are trying to classify.

Later layers are easier to train as adjusting their weights has a more immediate impact on the final result.

Guiding Principles for Fine Tuning

While there are no rules of thumb, these are some guiding principles to keep in mind:

- The more similar your data and problem are to the source data of the pre-trained network, the less intensive fine-tuning will be.
- If your data is substantially different in nature than the data the source model was trained on, Transfer Learning may be of little value.

CNN Architectures

LeNet-5

- Created by Yann LeCun in the 1990s
- Used on the MNIST data set.
- Novel Idea: Use convolutions to efficiently learn features on data set.

AlexNet

- Considered the “flash point” for modern deep learning
- Created in 2012 for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).
- Task: predict the correct label from among 1000 classes.
- Dataset: around 1.2 million images.

AlexNet developers performed data augmentation for training.

- Cropping, horizontal flipping, and other manipulations.

Basic AlexNet Template:

- Convolutions with ReLUs.
- Sometimes add maxpool after convolutional layer.
- Fully connected layers at the end before a softmax classifier.

VGG

Simplify Network Structure: has same concepts and ideas from LeNet, considerably deeper.

This architecture avoids Manual Choices of Convolution Size and has very Deep Network with 3x3 Convolutions.

These structures tend to give rise to larger convolutions.

This was one of the first architectures to experiment with many layers (More is better!). It can use multiple 3x3 convolutions to simulate larger kernels with fewer parameters and it served as ”base model” for future works.

Inception

Ideated by Szegedy et al 2014, this architecture was built to turn each layer of the neural network into further branches of convolutions. Each branch handles a smaller portion of workload.

The network concatenates different branches at the end. These networks use different receptive fields and have sparse activations of groups of neurons.

Inception V3 is a relevant example of an Inception architecture.

ResNet

Researchers were building deeper and deeper networks but started finding these issues:

In theory, the very deep (56-layer) networks should fit the training data better (even if they overfit) but that was not happening.

Seemed that the early layers were just not getting updated and the signal got lost (due to vanishing gradient type issues).

These are the main reasons why adding layers does not always decrease training error:

- Early layers of Deep Networks are very slow to adjust.
- Analogous to “Vanishing Gradient” issue.

- In theory, should be able to just have an “identity” transformation that makes the deeper network behave like a shallow one.

In a nutshell, a ResNet:

- Has several layers such as convolutions
- Enforces “best transformation” by adding “shortcut connections”.
- Adds the inputs from an earlier layer to the output of current layer.
- Keeps passing both the the initial unchanged information and the transformed information to the next layer.