Derek Lee Deep Learning Fall 2020

Professor Curro Quiz #5

1. Describe one of the experiments in ``Rethinking Generalization'' and what the implications were.

The authors tested deep neural networks on datasets where "the true labels were replaced by random labels", and the true images were replaced with Gaussian noise. Even though the images and the labels were random, the neural networks were able to achieve 0 training error. The authors conclude that convolutional neural networks can fit random noise. When the amount of noise was varied, the generalization error varied as well. As the level of noise was increased, the generalization error deteriorated. This shows that neural networks can still capture the remaining signal in the data, while also fitting to the noisy part "using brute force."

On CIFAR10, AlexNet and MLPs converged to zero training loss. On ImageNet, the Inception v3 model achieved a 95.20% accuracy, which is "very surprising for a million random labels from 1000 categories." The test accuracy was equivalent to random guessing, as expected.

2. Compare and contrast SqueezeNet with MobileNets.

SqueezeNets and MobileNets are designed as more efficient variants of convolutional neural networks. SqueezeNet uses 50x fewer parameters and is 510x smaller than AlexNet, while achieving a similar accuracy. MobileNets optimize for latency, which also yields small networks.

MobileNets use depthwise separable convolutions, which "factorize a standard convolution into a depthwise convolution and a 1x1 convolution called a pointwise convolution." The factorization drastically reduces computation and model size. MobileNet "uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy". Because MobileNets are small models, they require less regularization and data augmentation than large models, "because small models have less trouble with overfitting." The authors also note that it was important "to put very little or no weight decay… since [there] are so few parameters in them."

MobileNets introduce two parameters: width multiplier α and resolution multiplier ρ. The width multiplier reduces the number of channels, while the resolution multiplier reduces the

dimensionality of the image. These hyperparameters can reduce the computation. The authors show the log-linear dependence between accuracy and computation (Mult-Adds).

SqueezeNets use a building block called the Fire module. The Fire module has a squeeze layer, composed of 1x1 filters, which feeds into an expand layer, which as a mix of 1x1 and 3x3 filters. The architecture is designed based on three strategies: Replace 3x3 filters with 1x1 filters, decrease the number of input channels to 3x3 filters, and downsample late in the network. The first two decrease the number of parameters while attempting to preserve accuracy. The last strategy maximizes accuracy.

MobileNets and SqueezeNets both use a similar approach to produce smaller networks. They both alter convolutional layers, with MobileNets using depthwise separable convolutions, and SqueezeNets using various strategies to reduce computation by convolutional layers. However, MobileNets are optimized for latency, whereas SqueezeNets are optimized for model size.

3. What do you think of DenseNets?

DenseNets seem like a very interesting alternative to ResNets. DenseNets borrow the idea of creating a direct path to provide easier access to gradients, but do so on a larger scale, creating direct paths between, rather than within, layers. It also is seemingly more efficient than ResNets, since previous research has shown "many layers contribute very little and can in fact be randomly dropped during training." DenseNets are very parameter efficient, requiring "about 1/3 of the parameters as ResNet to achieve comparable accuracy." Instead of focusing on creating a wide or deep network, DenseNets focus on "feature reuse". Rather than combining features through summation, DenseNets concatenate them. DenseNets can have "very narrow layers, e.g. $k = 12$."

This "seemingly small modification" has a significant effect. DenseNets encourage feature reuse, as feature-maps in one layer can be used by all following layers, leading to model compactness. The authors show that features used in earlier layers are "directly used by deep layers throughout the same dense block." Layers consistently assign the least weight to the outputs of the transition layer. However, the classification layer uses the final feature-maps more than earlier feature-maps, "suggesting that there may be some more high-level features produced late in the network." It is possible that there is an implicit deep supervision, as each layer

receives “additional supervision from the loss function through the shorter connections.” There is at most two to three transition layers separating a hidden layer from the classification layer.