CS282A Lecture 10 Notes (09/27)

Matan Grinberg and Shruti Satrawada

September 2022

1 Residual Nets and their advantages

From last lecture, we recall the ResNet architecture.

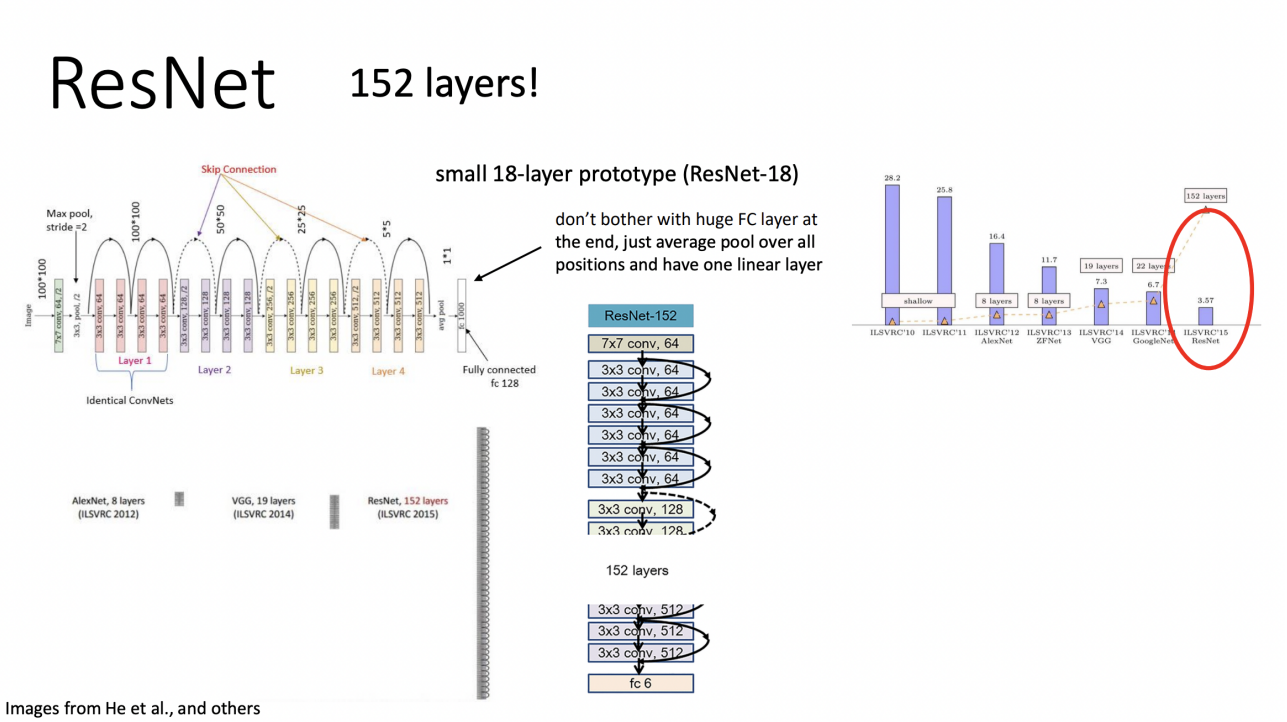

Figure 1: Overview of Residual Net architecture.

The modern convolutional neural network era was ushered in by the idea that each successive layer *modifies* the data in the pipeline, as opposed to completely transforming it.

We can also see how depth affects plain nets versus residual nets in Figure 2. An issue people found (that motivated this new residual architecture) was that in plain nets, introducing new layers created more error instead of creating more expressivity.

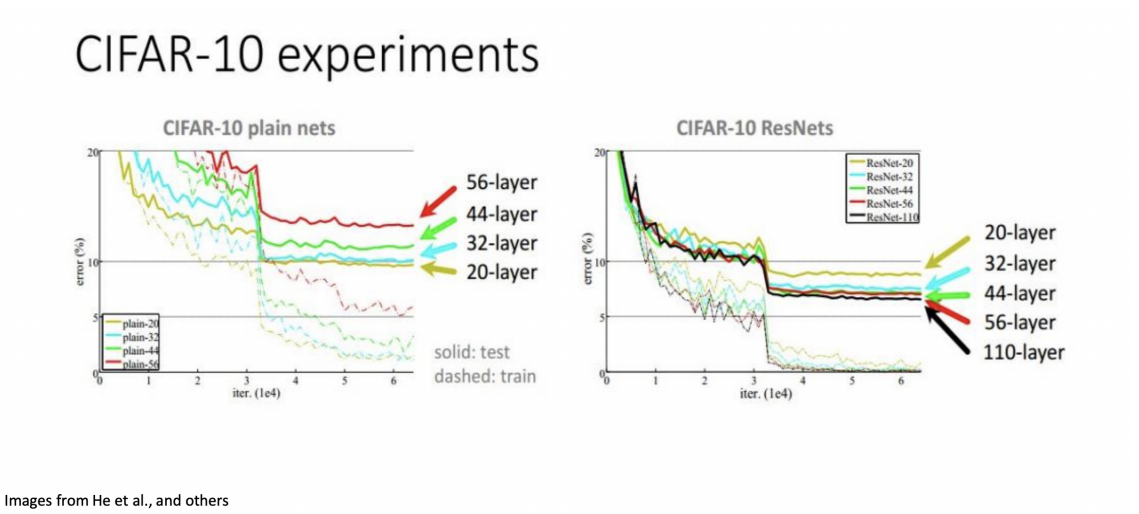

Figure 2: Effect of depth on error in CIFAR-10 experiments for plain nets and ResNets.

With this new residual net architecture (cf. Figure 3), each successive layer no longer completely transforms the data. Instead, through the use of the *skip-connection*, an identity operation is applied and added to the actual output of the layer. The effect of this is allowing the network to less dramatically modify the data in the pipeline, which in turn allows the depth of deep networks to more consistent learn salient features of the data.

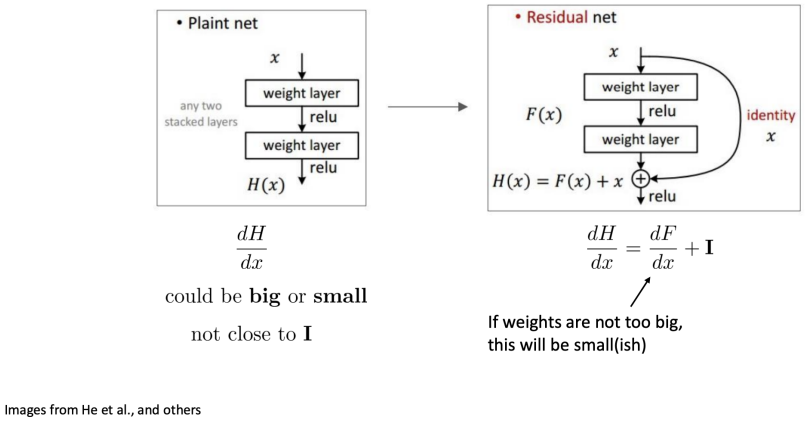

Figure 3: Difference between plain and residual architecture.

The derivative for a given layer of a residual net takes a form similar to that of Neural ODEs:

$$\frac{d}{ds}x(s) = \tilde{F}_s(x(s))\,, \tag{1}$$

where s is a fictitious "time" that signifies x going through the net and $\tilde{F}_s$ is learnable.

Now, as the number of these layers increases, do we eventually reach convergence? In other words, with enough layers, do things stop changing? We cannot in fact make general statements about the steady-state behavior of such deep networks. This is mainly due to the fact that we also cannot make any such guarantees about the behavior of ODEs of the form shown above. However, we can say if the residual blocks are small and get smaller in successive layers, then we would be able to make some guarantees about convergence.

2 ConvNeXt

2.1 Introducing ConvNeXt

A new architecture emerged that borrow from the success of transformers (without the actual transformer part). It differs in architecture from the residual net as shown in Figure 4.

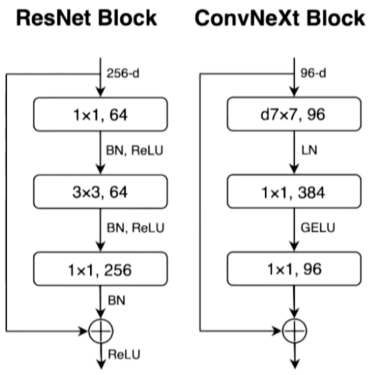

Figure 4: Residual net versus ConvNeXt. (Image from He et al.)

It is important to note that in ConvNeXt, instead of using the traditional batch normalization and ReLU activation, layer normalization and *Gaussian ReLU* (GeLU) are used. Furthermore, the 7×7 convolution is done within each channel, as opposed to across the channels (this is signified by the "d" in front of d7×7). Furthermore, ResNet has convolution operations taking place in the middle of the pipeline, whereas ConvNeXt has convolution in the beginning. This ultimately allows ConvNeXt to save a factor of 64 in parameters space, while also using and only one activation layer.

2.2 GeLU: Gaussian ReLU

We saw a new form of ReLU in the ConvNeXt structure called GeLU which stands for the Gaussian Error Linear Unit and is calculated by

$$x \cdot \Phi(x) \tag{2}$$

where $\Phi(x)$ represents the Gaussian CDF of x.

Theoretical Inspiration for GeLU: For ReLU we take the max(0, x), but why 0? Why not pick something else to act as our "gate" to decide whether the value x is passing through? Lets randomly draw a Gaussian, N. If x is greater than N, we pass the value through, and if not we return 0. GeLU is the expected value of this.

As can be seen in Figure 5, the GeLU curve is smoother than the ReLU curve, is non-convex and has a non-monotonic gradient.

This new formulation is discontinuous like ReLU.

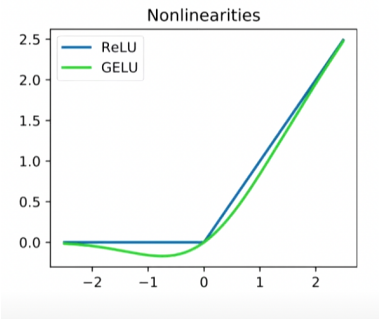

Figure 5: Graph comparing RELU versus GELU. Original Source: Wikipedia by Ringdongdang

2.3 Depthwise Convolution

In Depthwise Convolution, filters and inputs are broken into different channels, convoluted separately, and then concatenated.

In the ConvNeXt structure, Depthwise Convolution is used instead of typical convolution as it is less computationally demanding. In general nowadays, Depthwise Convolution is more commonly used than typical convolution. The difference between typical convolution and depthwise convolution is visualized in Figure 6

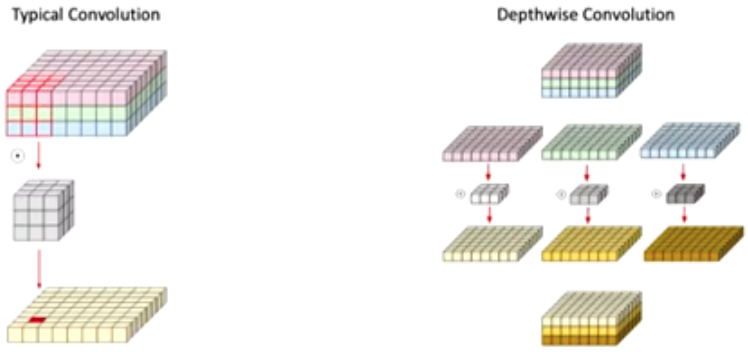

Figure 6: Regular Convolution versus Depthwise Convolution. Source: Lecture Slides.

2.4 Wrapping up ConvNeXt

Since ResNets in 2015, newer models like the 2022 ConvNeXt have improved performance as seen in Figure 7.

These newer models take bits and pieces from the old models. For example residual connections from ResNets are still common. However, there have been changes made as well, with depthwise (grouped) convolution being more common now than typical convolution.

ConvNeXt uses a cosine learning rate schedule, AdamW (Adam with weight decay), label smoothing, a type of dropout called Stochastic Depth Regularization, and aggressive data augmentation, all helping lead to higher accuracy.

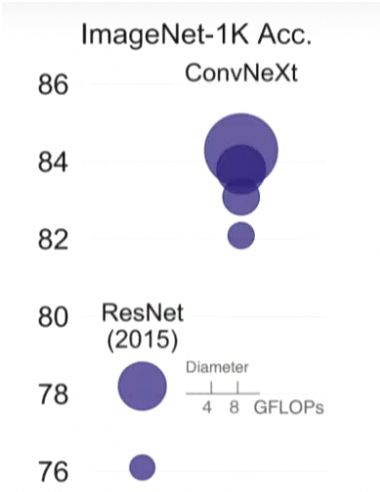

Figure 7: Performance of ResNet versus ConvNext on ImageNet-1K Acc. (Image from Liu et al, 2022)

3 Dropout

3.1 Dropout(basic)

Inspiration How can we simulate having an ensemble of diverse neural networks that we output the average of, like in Random Forest, in a method that's less costly? By utilizing dropout, there is essentially a slightly different neural network at each training step which provides an "ensemble-like" behavior.

Implementation

- During Training: While training on our minibatch, randomly "kill" certain units by setting them to zero before training. We will have a hyperparameter that provides the probability of "nulling" out a unit. When we "kill" a unit, all the attached weights do not get updated and we can think of the remaining weights "shaking" a bit to pick up the slack.
- Evaluation: The standard way evaluation is done after dropout is to use expected behavior.
  - Example: Lets say we're killing certain units with probability $\frac{1}{2}$. This means that our unit, x, is outputting 0 half the time and $ReLU(x)$ the other half of the time. We can calculate the expected behavior of our unit as $\frac{1}{2} \cdot ReLU(x) + \frac{1}{2} \cdot 0$. This essentially halves the output of our neuron during evaluation to account for us "nulling out" half the neurons while training.

Results of Dropout Dropout promotes diversity and redundancy in networks since at each training step we are essentially training a different neural network. It ensures that it isn't one unit's job to learn everything since that one unit can't always be relied on. It has a regularizing effect and is usually only used for MLPs (multi-layer perceptrons) on 1x1 blocks.

Does Dropout make training slower? No. Learning rate and dropout are trained together, so while dropout does reduce the size of our gradients, the learning rate can compensate for this.

Can Dropout hurt the performance? All regularization techniques have the ability to hurt performance since they shape the inductive bias. Depending on what we are modeling dropout can affect the performance differently. For a brief period of time, people thought that Normalization might negate the need for Dropout, but in practice having both performs better.

Mathematical Reasoning Behind Dropout In another lecture we discussed that we can think of our training algorithm seeing some version of the big singular values that are defined in the inductive bias. Generally for a matrix there are two ways you can have a large singular value. Either you have many different things pointing in the same direction, or you have one particular row or column that has a big value. Dropout drives us towards the direction of having lots of little things pointing in the right direction.

3.2 Stochastic Depth Regularization

Implementation During training randomly drop entire residual blocks. So during training some will be active and being trained while others are "gone". Very similar to Dropout but on a different scale.

3.3 Other Methods Similar to Dropout that Have Been Explored

- **Drop Connect**: Instead of "killing" certain units why not try "killing" weights? People have found that in practice this does not work as well.
- Another method that also adds the regularizing effect we see in Dropout is to multiply by random noise (in between 0 and 1) instead of multiplying by 0. Dropout tends to perform better in practice.
- What if we have logic behind what we turn off instead of randomly choosing? People have explored "killing" units more or less often based on what the size of their activation's tend to be as well as other similar ideas, but these have not been proven to be useful in practice so far.

4 Label Smoothing

4.1 Before Label Smoothing: One Hot Encoding

For Classic Cross-Entropy/Log-Loss Classification when we have a label "class 1", we represent this using one hot encoding. Since one hot encoding has an array of 0s with a 1 for the correct label as a goal instead of probabilities, it forces the model to try and be "super" confident. This means that the model will constantly be trying to get closer and closer and since with softmax we cannot actually reach a probability of 1 unless the input is infinity, the model will never be to reach the goal classification.

4.2 Label Smoothing

Label smoothing is an approach that fixes this by using probabilities in the goal array for our classification so our model can actually reach its goal.

We set our goal array y as follows:

$$y = [\frac{1 - \alpha \cdot k - 1}{k}, \frac{\alpha}{k}, \frac{\alpha}{k}, \frac{\alpha}{k}, ...]^T$$

where k is the number of classes and α is a hyperparameter

Reaching this goal y array is possible and in practice label smoothing does improve performance. We can think of the behavior as now being more similar to squared-error since they can both be satisfied.

Concern: Originally, people were concerned about this idea since in the real world some items are more similar than others. For example, a dog should be more similar to a cat than to the ocean. With that in mind, we can see that label smoothing tries to say that something is a dog but has an equal small probability of being either a cat or ocean when logically we would expect the probability for the cat to be higher.