Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

1 Citation

He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

`https://www.cv-foundation.org/openaccess/content_iccv_2015/papers /He_Delving_Deep_into_ICCV_2015_paper.pdf`

2 Abstract

We introduce the $f(x) = \mathbb{I}[y_i > 0]y_i + \mathbb{I}[y_i \leq 0]a_i$ parameterized ReLU (PReLU) unit to get superhuman (4.94% top-5 error) performance on ImageNet.

3 Introduction

Recent CNN improvements have built more powerful models and developed techniques to avoid over-fitting. ReLU is one such improvement because it makes convergence faster and the resulting accuracy better. Our PReLU learns the a_i value in our function.

4 Approach

ReLU has $a_i = 0$ and leaky ReLU (LReLU) has $a_i =$ constant. We can learn a_i for each channel in the layer (channel-wise) or one $a_i = a$ shared over all channels in the layer (channel-shared). If the objective function is $\mathcal{E}$, the gradient is (where summation is over all positions in feature map):

$$\frac{\partial \mathcal{E}}{\partial a_i} = \sum_{y_i} \frac{\partial \mathcal{E}}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i} = \sum_{y_i} \frac{\partial \mathcal{E}}{\partial f(y_i)} \mathbb{I}[y_i \leq 0]y_i \tag{1}$$

FOr the channel-shared case, we sum over the i (channels) and then the y_i (positions). We use the update rule: $\triangle a_i := \mu \triangle a_i + \epsilon \frac{\partial \mathcal{E}}{\partial a_i}$. We do NOT use a weight decay or that will push PReLU towards ReLU. We initialize with $a_i = 0.25$.

We train a 14-layer convnet on ImageNet and use PReLU (or ReLU). PReLU gives a 1.2% improvement in top-5 error (the channel-wise and channel-shared variants are comparable to each other). We find that a_i is high for the first layer (they are linear and are simply Gabor filters) but get smaller in deeper layer. Looking at the Fisher Information Matrix (FIM) during SGD, we see that PReLU pushes off-diagonal blocks closer to zero and improves conditioning, which is why it helps converge faster.

Most CNNs are initialized by drawing from a zero-mean Gaussian. For deeper models, like VGG, people tend to train the first several layers, then add more layers and train the rest. Another option is to add an auxiliary classifier in the middle of the network so loss can flow from there. Another option is the Xavier initialization (this assumes linear activation, which ReLU and PReLU are not).

We develop a new initialization method. First, assume that we can model layer l as $\boldsymbol{y}_l = W_l \boldsymbol{x}_l + \boldsymbol{b}_l$ and apply activation to get $\boldsymbol{x}_{l+1} = f(\boldsymbol{y}_l)$. If we assume that initial weights and input feature vector elements are i.i.d and that the weights are distributed with a zero-mean symmetric distribution, we can compute the variance of an L layer network as (where we assume y_L is an arbitrary element in the layer and y_1 is an arbitrary element in the first layer, and n_l is the length of $\boldsymbol{x}_l$):

$$Var[y_L] = Var[y_1]\left(\prod_{l=2}^{L} \frac{1}{2} n_l Var[w_l]\right) \tag{2}$$

To prevent this from exploding (thus diverging learning) or vanishing (thus stalling learning), we can set $\frac{1}{2} n_l Var[w_l] = 1 \; \forall l$. Thus, we can initialize the weights from a zero-mean Gaussian with standard deviation $\sqrt{1/n_l}$. We can also look at the gradient expression from last layer to first layer in a similar way and reach a similar initialization conclusion. We intialize biases with 0. You should normalize your input features, but if you don't we have another option for you (see paper). Comparing against Xavier initialization, our approach accounts for nonlinearities and does not stall when you train deep models.

For ImageNet, we did not get gains from using extremely deep (e.g. 30 layers) models.

5 Architectures and Implementation

Use a VGG-19 model as our basis and make a few small tweaks to improve the speed - we call this Model A. We add a few layers and get Model B (this is just marginally better than A, because depth doesn't seem to help). We make model B wider to get Model C. It takes 3-4 weeks to train our models.

We take random crops, flips, and color alteration to augment the dataset. We directly train the deep model (instead of training the first few layers and and then adding the later layers). We use dropout (50%). We train with SGD with momentum, cutting learning rate when validation error plateaus. Our minibatches have 128 images and we train for 80 epochs.

At test-time, we use the multi-view testing strategy as in the Spatial-Pyramid Pooling paper and also use a dense sliding window. Basically, we apply the conv layers, do pooling with Spatial Pyramid Pooling, and apply the fully-connected layers. We average the scores across all sliding window positions and all scales.

6 Experiments on ImageNet

We compare ReLU and PReLU on each of Model A, B, and C. PReLU converges faster and has lower validation error throughout training. It also gets better top-5 error. Model C with PReLU is best with 5.71% top-5 error. Model B is similar in accuracy to Model A. Thus, this shows that for already deep models, it is more important to increase width than increase depth. By ensembling models, we get 4.94% top-5 error, which is state of the art and also beats humans (5.1% top-5 error). This is because our model does better at fine-grained recognition (e.g. it can often tell the species of an animal in ImageNet, which a human cannot always do).

When we try our model on PASCAL VOC, however, it does not beat humans. We get good performance on PASCAL VOC object detection by using R-CNN and transferring our ImageNet pretrained models to the new task. On PASCAL VOC as well, PReLU beats ReLU.