Dropout rate guidance for hidden layers in a convolution neural network

Asked 7 years ago Modified 6 years, 11 months ago Viewed 26k times

14

I am currently building a convolution neural network to play the game 2048. It has convolution layers and then 6 hidden layers. All of the guidance online mentions a dropout rate of ~50%. I am about to start training but am concerned that 50% dropout on each of the 6 layers is a bit overkill and will lead to under-fitting.

I would greatly appreciate some guidance on this. What do you guys recommend as a starting point on dropout? I would also love to understand why you recommend what you do.

machine-learning neural-network conv-neural-network convolution recurrent-neural-network

Share Follow

edited Dec 24, 2017 at 10:40

Maxim
53.7k ● 27 ● 158 ● 211

asked Dec 19, 2017 at 17:44

user88383
197 ● 1 ● 1 ● 9

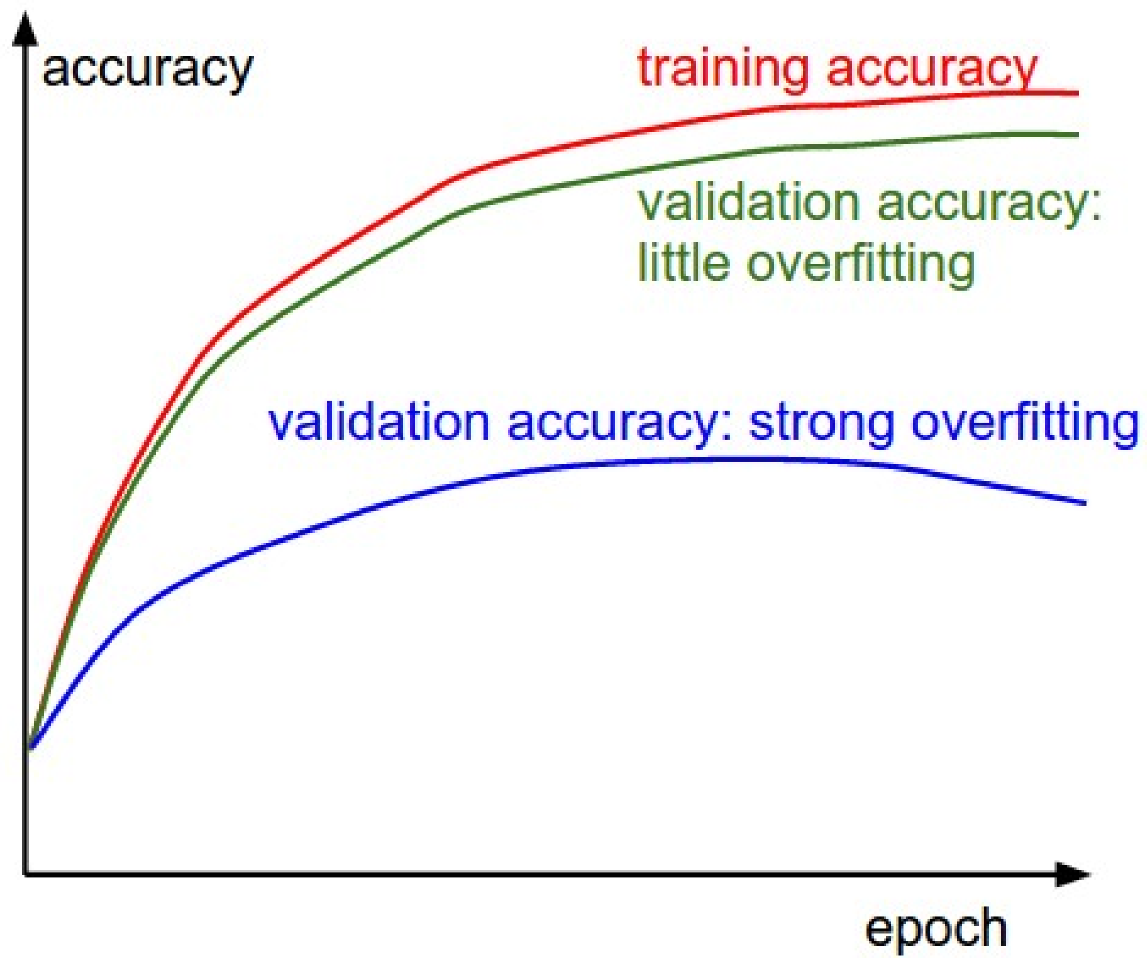
1 Answer

Sorted by: Highest score (default)

27

First of all, remember that dropout is a technique to *fight overfitting* and improve neural network generalization. So the good starting point is to focus on training performance, and deal with overfitting once you clearly see it. E.g., in some machine learning areas, such as reinforcement learning, it is possible that the main issue with learning is lack of timely reward and the state space is so big that there's no problem with generalization.

Here's a very approximate picture how overfitting looks like in practice:



By the way, dropout isn't the only technique, the latest convolutional neural networks tend to prefer batch and weight normalization to dropout.

Anyway, suppose overfitting is really a problem and you want to apply specifically dropout. Although it's common to suggest `dropout=0.5` as a default, this advise follows the recommendations from the original Dropout paper by Hinton at al, which at that time was focused on fully-connected or dense layers. Also the advise implicitly assumes that the researches does hyper-parameter tuning to find the best dropout value.

For convolutional layers, I think you're right: `dropout=0.5` seems too severe and the research agrees with it. See, for example, "Analysis on the Dropout Effect in

Convolutional Neural Networks" paper by Park and Kwak: they find that much lower levels `dropout=0.1` and `dropout=0.2` work better. In my own research, I do Bayesian optimization for hyper-parameter tuning (see this question) and it often selects gradual increase of drop probability from the first convolutional layer down the network. This makes sense because the number of filters also increases, so does the chance of co-adaptation. As a result, the architecture often looks like this:

- CONV-1: `filter=3x3`, `size=32`, dropout between `0.0-0.1`
- CONV-2: `filter=3x3`, `size=64`, dropout between `0.1-0.25`
- ...

This does perform well for classification tasks, however, it's surely not a universal architecture and you should definitely cross-validate and optimize hyper-parameters for your problem. You can do that via simple random search or Bayesian optimization. If you choose Bayesian optimization, there're good libraries for that, such as this one.

Share Follow

answered Dec 24, 2017 at 9:44

Maxim
53.7k ● 27 ● 158 ● 211

1 Thank you so much for the detailed feedback - I greatly appreciate it. I also greatly appreciate the references!
– user88383 Dec 24, 2017 at 21:48