# Don’t Use Dropout in Convolutional Networks

**<= Previous post**
**Next post =>**

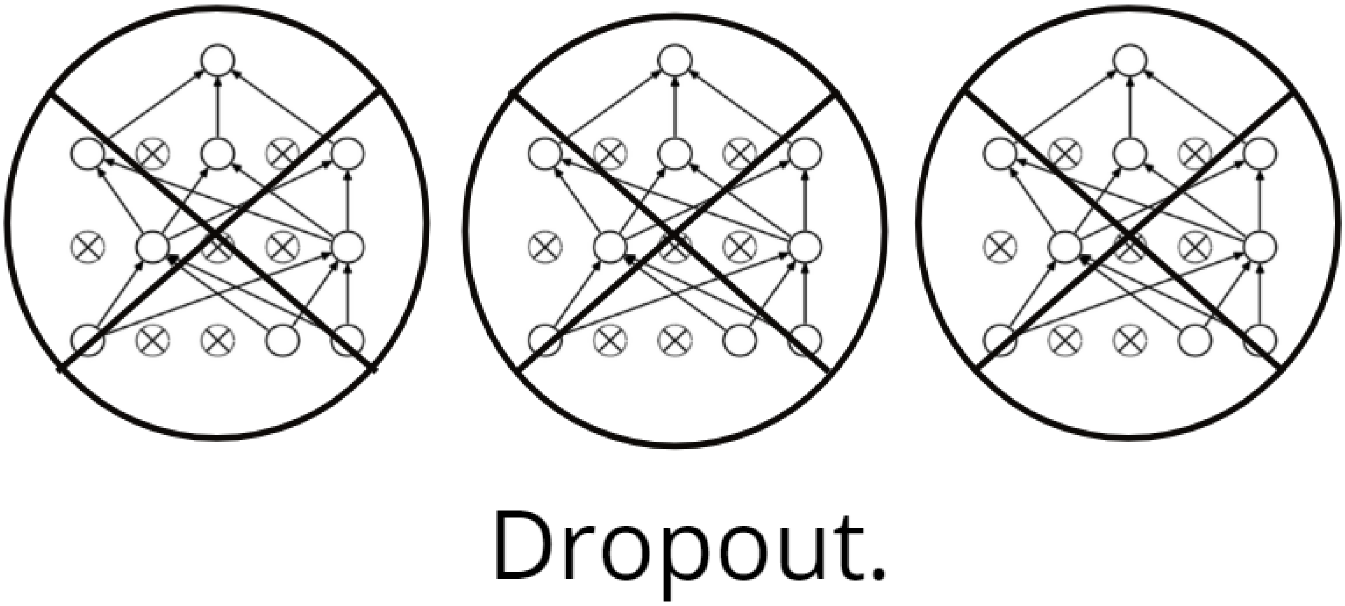
Like 84 Share 84 Tweet Share 24

Tags: Convolutional Neural Networks, Dropout, Keras

If you are wondering how to implement dropout, here is your answer - including an explanation on when to use dropout, an implementation example with Keras, batch normalization, and more.

comments

**By Harrison Jansma.**



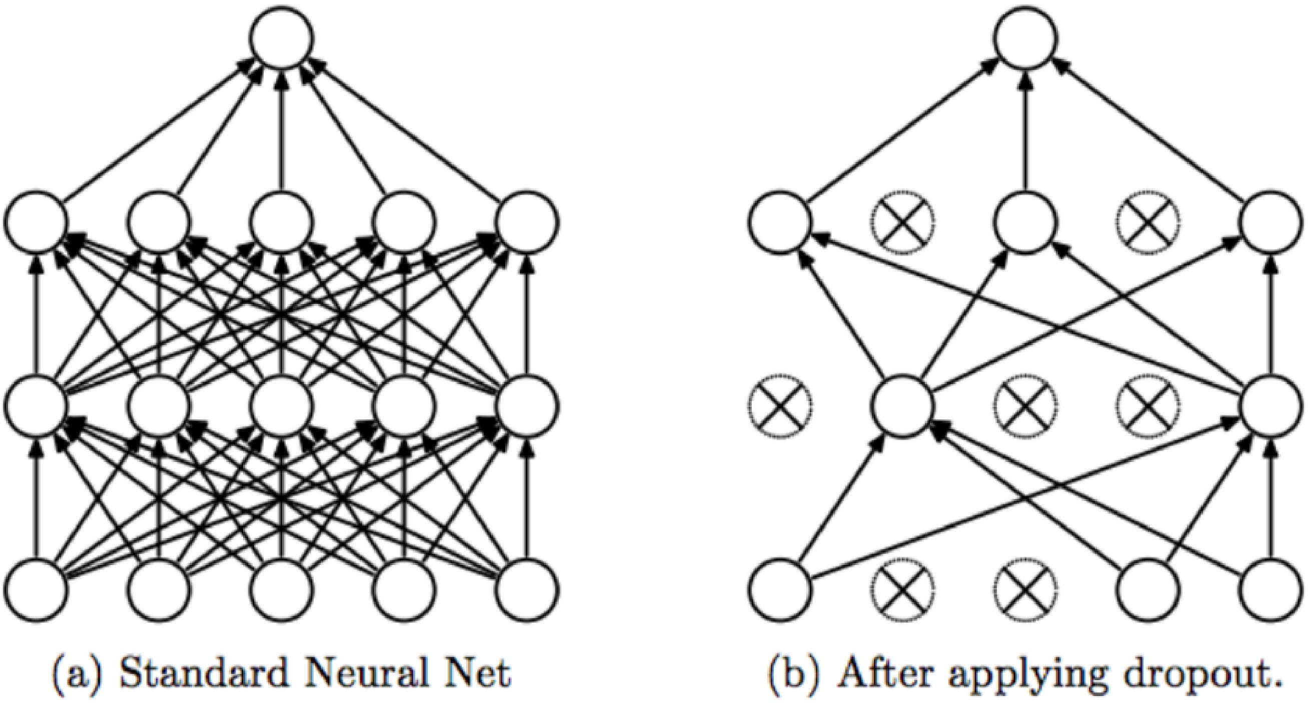
I have noticed that there is an abundance of resources for learning the what and why of deep learning. Unfortunately when it comes time to make a model, their are very few resources explaining the when and how.

I am writing this article for other data scientists trying to implement deep learning. So you don’t have to troll through research articles and Reddit discussions like I did.

In this article you will learn why dropout is falling out of favor in convolutional architectures.

## Dropout

If you are reading this, I assume that you have some understanding of what dropout is, and its roll in regularizing a neural network. If you want a refresher, read this post by Amar Budhiraja.



Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Generally, we only need to implement regularization when our network is at risk of overfitting. This can happen if a network is too big, if you train for too long, or if you don’t have enough data.

If you have fully-connected layers at the end of your convolutional network, implementing dropout is easy.

## Keras Implementation

```
keras.layers.Dropout(rate, noise_shape=None, seed=None)
```

- Start with a dropout rate of 0.5 and tune it down until performance is maximized. (Source.)

## Example:

```
model=keras.models.Sequential()
model.add(keras.layers.Dense(150, activation="relu"))
model.add(keras.layers.Dropout(0.5))
```

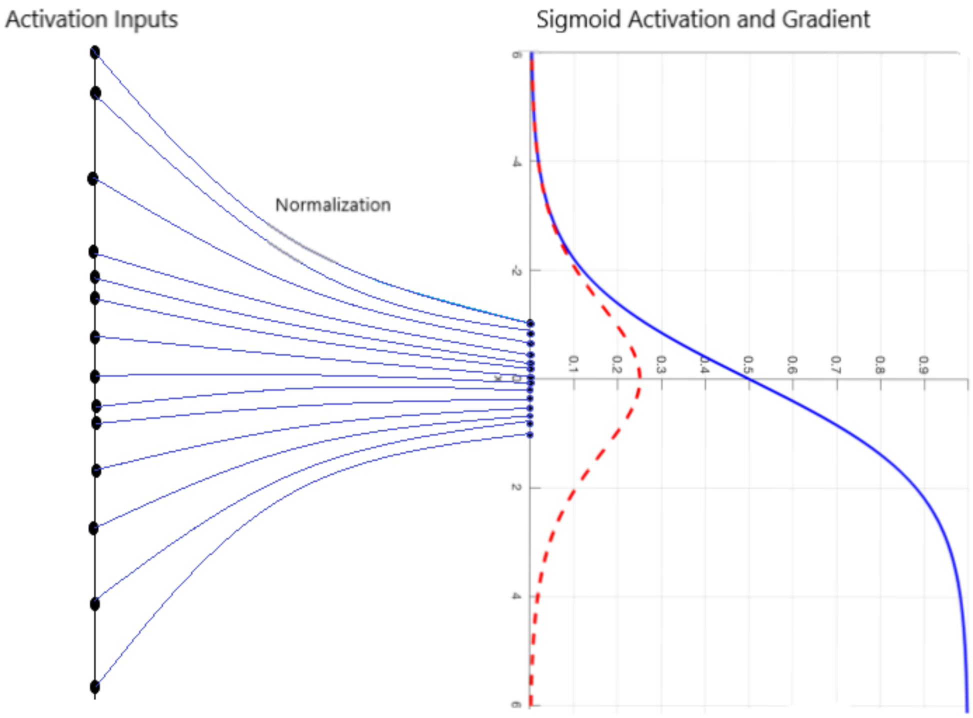
Note that this only applies to the fully-connected region of your convnet. For all other regions you should not use dropout.

Instead you should insert batch normalization between your convolutions. This will regularize your model, as well as make your model more stable during training.

## Batch Normalization.

Batch normalization is another method to regularize a convolutional network.

On top of a regularizing effect, batch normalization also gives your convolutional network a resistance to vanishing gradient during training. This can decrease training time and result in better performance.



Batch Normalization Combats Vanishing Gradient

## Keras Implementation

To implement batch normalization in Keras, use the following:

```
keras.layers.BatchNormalization()
```

When constructing an convolutional architecture with batch normalization:

- Insert a batch normalization layer between convolution and activation layers. (Source.)
- There are some hyperparameters you can tweak in this function, play with them.

You can also insert batch normalization after the activation function, but in my experience both methods have similar performance.

## Example:

```
model.add(Conv2D(60,3, padding = "same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
```

## Batch normalization replaces dropout.

Even if you don’t need to worry about overfitting there are many benefits to implementing batch normalization. Because of this, and its regularizing effect, batch normalization has largely replaced dropout in modern convolutional architectures.

> *“We presented an algorithm for constructing, training, and performing inference with batch-normalized networks. The resulting networks can be trained with saturating nonlinearities, are more tolerant to increased training rates, and often do not require Dropout for regularization.” - Ioffe and Svegedy 2015*

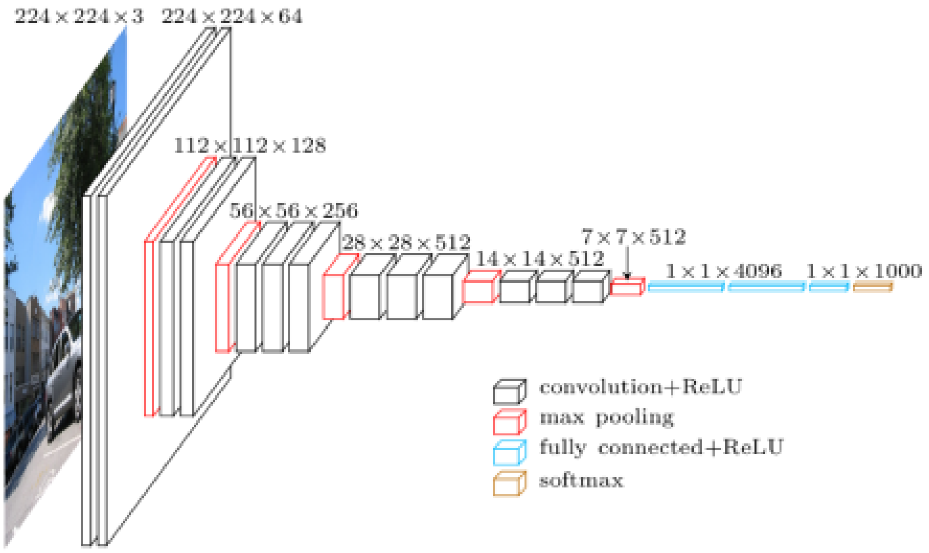
As to why dropout is falling out of favor in recent applications, there are two main reasons.

**First, dropout is generally less effective at regularizing convolutional layers.**

The reason? Since convolutional layers have few parameters, they need less regularization to begin with. Furthermore, because of the spatial relationships encoded in feature maps, activations can become highly correlated. This renders dropout ineffective. (Source.)

**Second, what dropout is good at regularizing is becoming outdated.**

Large models like VGG16 included fully connected layers at the end of the network. For models like this, overfitting was combatted by including dropout between fully connected layers.



VGGNet and it’s Dense Head

Unfortunately, recent architectures move away from this fully-connected block.

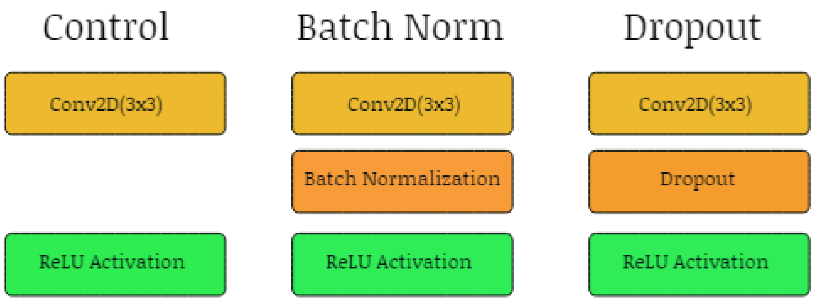
By replacing dense layers with global average pooling, modern convnets have reduced model size while improving performance.

I will write another post in the future detailing how to implement global average pooling in a convolutional network. Until then, I recommend reading the ResNet paper to get an idea of GAPs benefits.

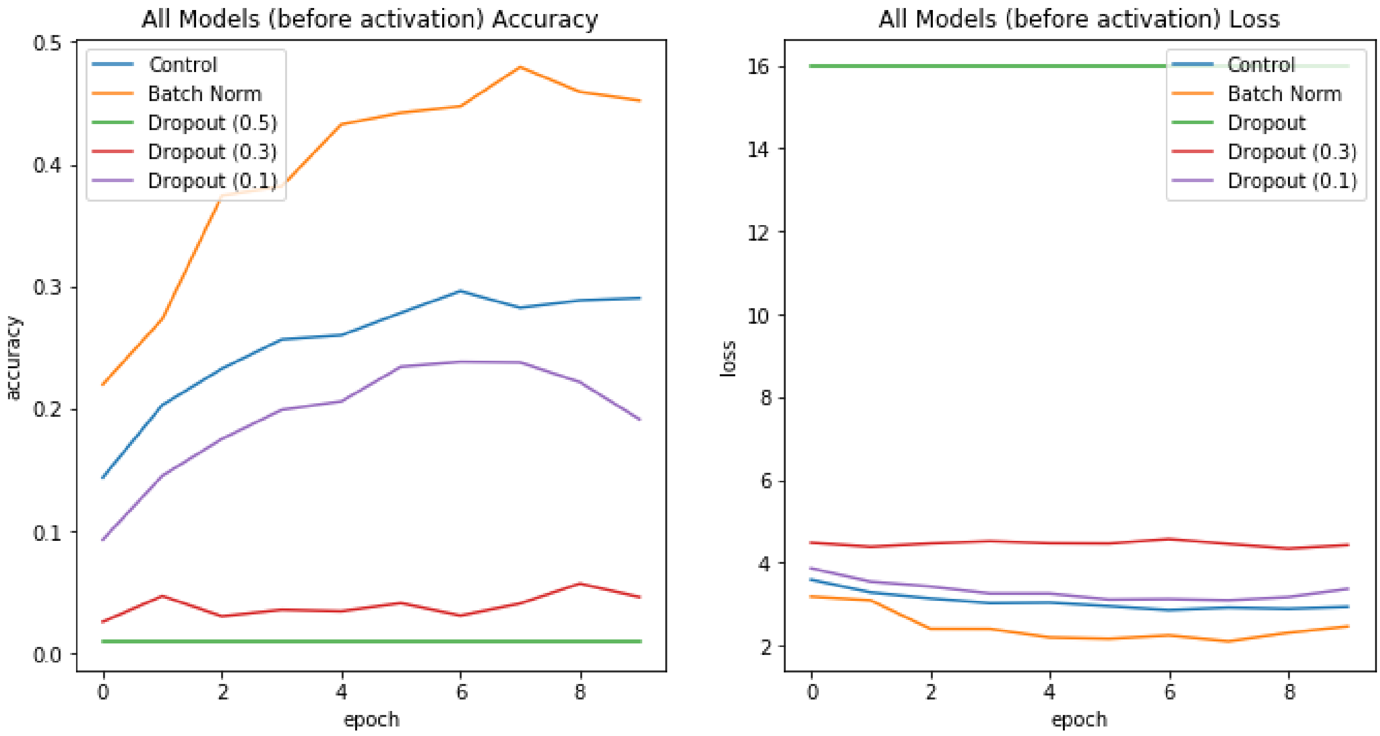
## An Experiment

I created an experiment to test whether batch normalization reduces generalization error when inserted between convolutions. (link)

I constructed 5 identical convolutional architectures, and inserted either dropout, batch norm, or nothing (control) between convolutions.



By training each model on the Cifar100 dataset I achieved the following results.



The high performance of the batch-normalized model supports the claim that batch normalization should be used between convolutions.

Furthermore, dropout should not be placed between convolutions, as models with dropout tended to perform worse than the control model.

For more information check out the full write-up on my GitHub.

## Takeaways

If you were wondering whether you should implement dropout in a convolutional network, now you know. Only use dropout on fully-connected layers, and implement batch normalization between convolutions.

If you want to learn more about batch normalization, read this:

https://towardsdatascience.com/intuit-and-implement-batch-normalization-c05480333c5b

**Bio**: Harrison Jansma is a Masters Student in Computer Science at the University of Texas at Dallas. Harrison is fascinated by computer vision, machine learning, and back-end development.

Original. Reposted with permission.

**Related:**

- [Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks]
- [Deep Learning Tips and Tricks]
- [Inside the Mind of a Neural Network with Interactive Code in Tensorflow]

**What do you think?**

8 Responses

Upvote Funny Love Surprised Angry Sad

**2 Comments** **KDnuggets** 1 **Login**

♡ **Recommend** 1 Tweet f Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**Kyle Vrooman** • a year ago

Densenet, Wide Residual Nets, Xception, and other high complexiy models use Dropout. A related method called Cutout (which is simply a block dropout on the input image) is SOTA on several use cases. Batch Normalization can either be degrading to model performance in many cases ( domain adaption ) or computational intensive in comparison to other options ( data augmentation, weight regularization/decay, dropout, SELU activations, spectral normalization). There are always cases for or against.

Plus, while generic dropout is useful, randomly dropping entire activations/features works very well when spatial correlation exists among the nodes/pixels... it's a simple parameter tweak in the Keras Dropout layer that performs well on many convolution models

```
model.add(Dropout((0.1, noise_shape=[batch_size, channel_size, 1, 1])))
```

Plus, as a general rule, a model shouldn't use regularization unless it's overfitting the data/ approaching 100% accuracy due to large model capacity. The relatively simple example model with base accuracy at 30% isnt the best use case to showcase dropout and will of course make it look bad.

1 ^ | ˅ • Reply • Share ›

**Thomas Ott** • a year ago

Thanks for this write up. I've relied on dropouts quite a bit in the past. I'm going to investigate batch normalization here.

^ | ˅ • Reply • Share ›

[<= Previous post]
[Next post =>]

## Top Stories Past 30 Days

### Most Popular

1. How to Become a (Good) Data Scientist — Beginner Guide
2. Everything a Data Scientist Should Know About Data Management
3. The Last SQL Guide for Data Analysis You'll Ever Need
4. How to select rows and columns in Pandas using [ ], .loc, iloc, .at and .iat
5. 10 Great Python Resources for Aspiring Data Scientists
6. 10 Free Top Notch Natural Language Processing Courses
7. Which Data Science Skills are core and which are hot/emerging ones?

### Most Shared

1. How YouTube is Recommending Your Next Video
2. Everything a Data Scientist Should Know About Data Management
3. How to Become a (Good) Data Scientist – Beginner Guide
4. 10 Free Top Notch Natural Language Processing Courses
5. The 4 Quadrants of Data Science Skills and 7 Principles for Creating a Viral Data Visualization
6. Artificial Intelligence: Salaries Heading Skyward
7. Introduction to Natural Language Processing (NLP)

## Latest News

- Top October Stories: How to Become a (Good) Data Scient...
- What is Data Science?
- Understanding Boxplots
- Orchestrating Dynamic Reports in Python and R with Rmd ...
- 3 Reasons to attend Data Natives, 25-26 November, Berlin
- Monitoring Models at Scale

### Top Stories
### Last Week

## Most Popular

1. 5 Statistical Traps Data Scientists Should Avoid

2. Everything a Data Scientist Should Know About Data Management
3. How to Become a (Good) Data Scientist — Beginner Guide
4. How to select rows and columns in Pandas using [ ], .loc, iloc, .at and .iat
5. How to Build Your Own Logistic Regression Model in Python
6. How YouTube is Recommending Your Next Video
7. Top Machine Learning Software Tools for Developers

## Most Shared

1. Top Machine Learning Software Tools for Developers
2. Why is Machine Learning Deployment Hard?
3. Data Sources 101
4. 5 Statistical Traps Data Scientists Should Avoid
5. How to Extend Scikit-learn and Bring Sanity to Your Machine Learning Workflow
6. What is Machine Learning on Code?
7. Build an Artificial Neural Network From Scratch: Part 1

KDnuggets Home » News » 2018 » Sep » Opinions » Don’t Use Dropout in Convolutional Networks ( 18:n34 )