Adam: A Method for Stochastic Optimization (2014). Kingma & Ba,

https://arxiv.org/abs/1412.6980

This paper presents a method for stochastic gradient descend to correct noise that appears from randomly sampling data or from the optimization function itself. It uses first and second order moments (mean and un-centered variance) for that to calculate step size at each loop of the SGD, that is instead of just randomly taking gradient for some mini-batch (or train example) and changing parameters, this algorithm uses previous gradients through moments to calculate new parameters. Hence, its name - ADAptive Moment estimation. The moments calculate expected values at certain time using predefined parameters so that when there is little noise it takes larger steps rather than small steps when larger noise, which is the case near optimum. The method requires little computation and memory resources (several additional arithmetic operations and variables). It proposes other algorithms to enhance ADAM based on other moments. Experiments conducted for three popular objective functions, namely logistic regression, multi-layer neural networks and convolutional neural networks, show that Adam outperforms other algorithms, i.e. it converges faster and better.

Srivastava et al., Dropout: A Simple Way to Prevent Neural Networks from Overfitting (2014)

https://jmlr.org/papers/v15/srivastava14a.html

Dropout is a method to train large neural networks to solve the problem with overfitting. When training whole large NN it is common that they become overfit as the data comes limited and with noise. The best method to overcome it is to train many NN with different hyperparameters and use all of them at test time by averaging their predictions by how good they were at train time. But this is not feasible as compute is limited. Dropout proposes to 'drop' units of NN and their connections for each training example, thus creating many thin NN that share weights. At the test time no dropout happens and the whole NN is used, thus having the averaging effect that solves the overfit problem. Input units have close to 1.0 probability of staying for training, while other units have about 0.5. Cons of dropout is that it increases training time, typically by 2-3 times. So the trade-off between overfitting and training time created. The paper gives formally describes the model, gives algorithm, demonstrates usage of dropout on different domains, has section on how dropout relates to hyperparameters, gives advice how to implement it, and other.