Adam vs SGD Optimizers

For a brief of introduction of the optimizers, refer
https://deeptek.atlassian.net/browse/CONTENT-12

Challenges faced while using Gradient Descent and its variants:

1. Choosing a proper learning rate can be difficult.
   Low LR- Baby steps, slow convergence
   High LR: Might diverge and never give optimal weights
2. Additionally, the same learning rate applies to all parameter updates .If our data is sparse and our features have very different frequencies, we might not want to update all of them to the same extent, but perform a larger update for rarely occurring features.
3. Another key challenge of minimizing highly non-convex error functions common for neural networks is avoiding getting trapped in their numerous sub-optimal local minima. Actually, Difficulty arises in fact not from local minima but from saddle points ,i.e. points where one dimension slopes up and another slopes down. These saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape, as the gradient is close to zero in all dimensions.

How is Adam beneficial ?

Adam works well in practice and compares favorably to other adaptive learning-method algorithms as it converges very fast and the learning speed of the Model is quiet Fast and efficient and also it rectifies every problem that is faced in other optimization techniques such as vanishing Learning rate , slow convergence or High variance in the parameter updates which leads to fluctuating Loss function

Where does Adam fall short?

Primary findings from the paper(The Marginal Value of Adaptive Gradient Methods in Machine Learning):
(i) Adaptive methods find solutions that generalize worse than those found by non-adaptive methods.
(ii) Even when the adaptive methods achieve the same training loss or lower than non-adaptive methods, the development or test performance is worse.
(iii) Adaptive methods often display faster initial progress on the training set, but their performance quickly plateaus on the validation set.
(iv) Though conventional wisdom suggests that Adam does not require tuning, we find that tuning the initial learning rate and decay scheme for Adam yields significant improvements over its default settings in all cases

Brief comparison of Adam and SGD:

Adaptive (Adam, AdaGrad)	**SGD**
Optimizes Faster	Optimizes Slower
Works well for sparse datasets	Generalizes better

Improving Generalization Performance by Switching from Adam to SGD

1) One disadvantage of SGD is that it scales the gradient uniformly in all directions; this can be particularly detrimental for ill-scaled problems. This also makes the process of tuning the learning rate α circumstantially laborious.
2) To correct for these shortcomings, several adaptive methods have been proposed which diagonally scale the gradient via estimates of the function's curvature.
3) Adam outperforms SGD in both training and generalization metrics in the initial portion of the training, but then the performance stagnates

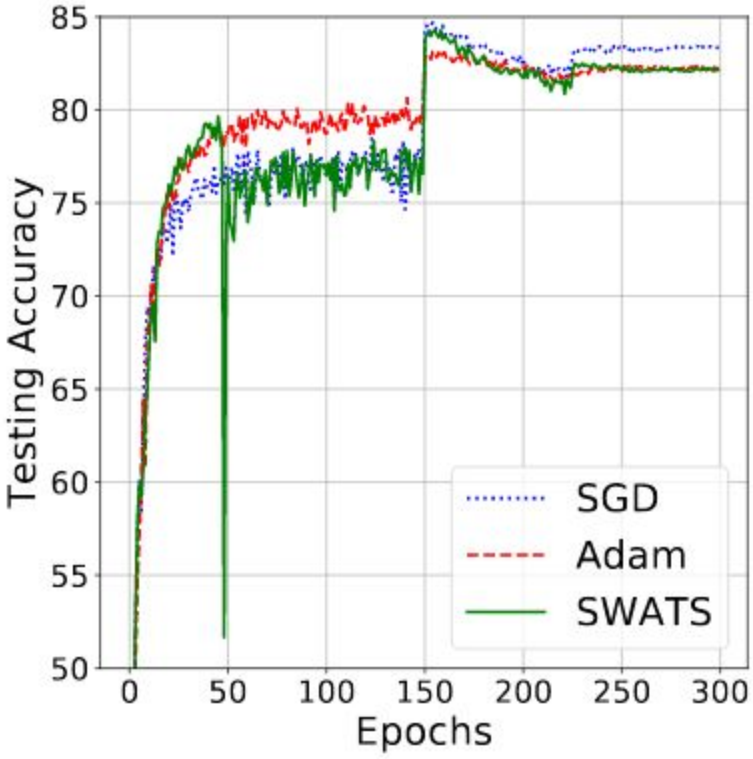

Figure 5. Numerical experiments comparing SGD(M), Adam and SWATS with tuned learning rates on the ResNet-18 architecture on the Tiny-ImageNet data set.

Other Reference Links:

https://ruder.io/optimizing-gradient-descent/

https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/

https://datascience.stackexchange.com/questions/30344/why-not-always-use-the-adam-optimization-technique/30347