MLP Coursework 1: Learning Algorithms and Regularization

s1149359

Abstract

In this report I explored the implementation and performance of RMS-Prop and Adam learning algorithms based on the multi-layers neural network. Further I implemented a learning rate scheduler for stochastic gradient descent and Adam to explore the relationship between learning rate and learning performance. Also I compared the learning performances between Adam with L2 regularization and weight decay. Finally, the learning scheduler was also applied to the Adam with weight decay to further explore the effect about sets of hyperparamters to the final performance. Some key comparisons and conclusions based on my experiments are included in this report.

1. Introduction

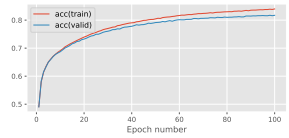
The multi-layer neural network has the strong ability to learn highly abstract representation from the input at the hidden layers, which can be subsequently applied for classification and regression tasks. A learning procedure for neural network can be simply considered as an optimization procedure that fits the parameters to data. The gradient-based algorithm is usually employed to iteratively update the parameters to reduce the value of objective function in neural network. Hence, a well-chosen optimization algorithm plays a key role in achieving state of the art performance when we train the multi-layer neural network. The importance of gradient-based algorithms in neural network motivates us to explore the implementations and performances of adaptive learning rate optimization algorithms such as Adam and RMS-Prop algorithms in the multi-layer neural network.

If the neural network can be trained well, the cost function should be more and more close to the maximization or minimization as the learning procedure goes . The learning step after every parameter update should gradually decay so that the cost function can finally reach the global optimal value (or local optimal value for most non-convex cases). Hence, learning rate, a key hyper-parameter controlling the learning step, is crucial to the learning success. In theory, in many circumstances a decayed learning rate would achieve better performance than a fixed learning rate in the gradient-based algorithm. A typical successful case of algorithms with a controllable learning rate is stochastic gradient descent with warm restarts (SGDR) Loshchilov & Hutter, which can periodically decay the learning rate. Hence, we are highly motivated to apply a learning rate scheduler which can periodically control learning rate to Adam to further tune the learning rate adaptively so that we could explore the how the performance correspondingly changes as the learning rate changes. In addition, L_2 regularization and weight decay are two effective regularization methods which could usually help improve the learning performance.Loshchilov and Hutter pointed out the difference between the two mechanisms, and demonstrated weight decay usually generalizes better than L_2 regularization (Loshchilov et al.). Being inspired by Loshchilov and Hutter's work our following experiments would also explore the implementation of these two regularization methods in Adam.

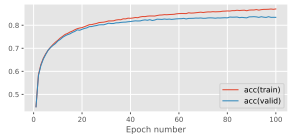
In experiments we used EMNIST Balance dataset, which extends MNIST by including images of handwritten letters as well as handwritten digits. This dataset has already been split into training set(100000 data points), validation set (15800 data points) and test set (15800 data points). Then for accelerating the training speed we partition the training set into multi minibatches, and each of minibatch includes 1000 data points. So the size of input for our multi layers neural network is 1000×784.

2. Baseline systems

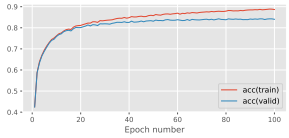
It is almost always a good idea to implement the simplest possible algorithm to give a solution for machine learning tasks. Then this solution can be considered as a baseline to approximately check how good or bad our new algorithms perform. If our new algorithms cannot even beat the baseline then we should consider whether our algorithms are ill-designed or ill-implemented. Basis on this idea, we first implement the basic stochastic gradients descent (SGD) algorithm with a fixed leaning rate for multi-layer neural network. Any explicit regularization is not also included for SGD. To slightly explore the effect of the number of layers to learning performance, we implemented neural networks with 2-5 hidden layers respectively, then compared the accuracy rate based on the validation set. Finally we reported the test accuracy rates for each type of multi-layer neural network based on the best set of hyper-parameters that we choose from the validation set. Figure 1 shows the trends of accuracy rate curve changing with respect to the epoch for neural network with 2-5 hidden layers. In detailed I set two hyperparameters: epoch=100, fixed learning rate=0.01 for four types of neural network. I also set an early stopping scheme for storing the optimal sets of learned parameters and avoiding overfitting. To be more specific, the training procedure would immediately stop if



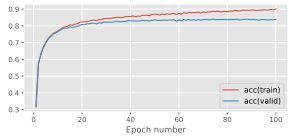
(a) Neural network with 2 hidden layers



(b) Neural network with 3 hidden layers

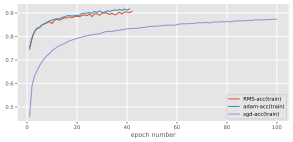


(c) Neural network with 4 hidden layers

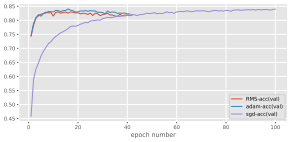


(d) Neural network with 5 hidden layers

Figure 1: Accuracy rate curves for multi-layer neural network



(a) Accuracy rate curve on training set



(b) Accuracy rate curve on validation set

Figure 2: Accuracy rate curves for RMSProp, Adam and SGD

the accuracy rate on the validation set keeps less than the highest value after a fixed training epoches. Here I set the fixed training epchoches as 25. As can be seen from the figure 1, the accuracy rates on training set slightly go better as the hidden layers are increase. This trend matches the theory assumption that the more complicated model would have better training performance. However the story is little different in validation set. The accuracy rate on the validation is not increased any more when the number of epoch is close to 100 for neural network with 4 and 5 hidden layers. Meanwhile, the highest accuracy rates on the validation set are also approximately equal for both two types of network. Finally, accuracy rates of the 4 types of neural networks with their optimal setting of hyperprameter on the test set is following: neural network with 2 hidden layers: 81.26%:, neural network with 3 hidden layers: 82.60%,neural network with 4 hidden layers 82.79%:,neural network with 5 hidden layers: 82.81%.

3. Learning algorithms – RMSProp and Adam

One of major shortcomings of SGD is that SGD cannot tune the learning rate according to learning procedure. The fixed learning rate might be suitable for some period of learning process but not the whole learning procedure. Meanwhile, the gradients after every update step fluctuate much, which could cause a slow convergence speed. In order to overcome the disadvantages of SGD we implemented two adaptive learning rate optimization algorithms RMSProp and Adam based on the multi-layer neural network in the baseline and explored how the changeable learning rate could affect the learning performance. The RMSProp updates the parameters by averaging the exponentially weighted moving of history gradients. Hence, every update step would be more smooth than SGD. Meanwhile, as the exist of uncenter second-order moments $(\rho\mathbf{r} + (1-\rho)\mathbf{g} \odot \mathbf{g})$, RMSProp would discard the effect of history gradients from the extreme past, which can converge more rapidly. Basis on the RMSProp, Adam combines with the estimate of the first-order moments of the gradients $(\rho_1\mathbf{s} + (1-\rho_1)\mathbf{g})$. Meanwhile, Adam includes a bias correction to the estimates of both the first order and second order moments. For the estimates of second order moments without correction in RMSProp, the bias of estimate would be high during first several batches training. To be more specific, at the beginning of the training item $(1-\rho_2)\mathbf{g} \odot \mathbf{g}$ makes the estimate value extreme small as the initial estimate value is set to zero in RMSProp. By contrast, the bias correction in Adam would correct the bias by dividing 1-ρ^t for both moments in early training. The algorithmic pseudocodes for RMSProp and Adam are present in algorithm 1 and algorithm 2. Hence, in section 3 we would explore how an adaptive learning rate affects performance, and how much the exist of first order estimate and bias correction item could improve the learning performance. According to the book of deep learning(Goodfellow et al.), the setting of hyper parameter is following: RMSProp: ρ=0.9, step size ϵ=0.001 , δ=10^{-6} ; Adam: step size ϵ= 0.001, ρ_1=0.99, ρ_2=0.999, δ=10^{-8}. The early stopping scheme also is implemented in this experiment. Figure 2 shows the trends of accuracy rate curve changing with respect to the epoch for RMSProp, Adam and SGD. As the use of early stopping the accuracy rate curves stop before 100 epoches for both Adam and RMSProp. As can be seen from the figure, compared to SGD, one of the most remarkable advantages of both RMSProp and Adam is the much faster convergence speed. It proves the adaptive learning rate algorithms could automatically tune the learning rate according to the current learning procedure, which could help model reach its some local optimal point in early training. In addition, it is worth to notice that in training set both algorithms achieve higher while in the validation set the final learning performance does not have a significant improvement. This phenomenon suggests us that we could try a learning rate scheduler to help Adam and RMSProp further tune the learning rate even though they already reach a local optimal point. Such a learning rate scheduler could utilize the advantage of rapid convergence of Adam and RMSProp to force the algorithms to explore multiple local optimal solutions which has a potential way to improve the final learning performance. Finally, accuracy rates of RMSProp with their optimal setting of hyperprameter on the test set is: 82.57%, and accuracy rates of Adam with their optimal setting of hyperprameter on the test set is: 82.79%.

Algorithm 1 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilise division by small numbers
Initialise accumulation variables $\mathbf{r} = 0$
while stopping criterion not met **do**
  Sample a mini batch of m samples from the training set $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
  Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$.
  Accumulate squared gradient: $\mathbf{r} \leftarrow \rho\mathbf{r} + (1-\rho)\mathbf{g} \odot \mathbf{g}$.
  Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta+\mathbf{r}}} \odot \mathbf{g}$. ($\frac{1}{\sqrt{\delta+\mathbf{r}}}$ applied element-wise)
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

Algorithm 2 The Adam algorithm

Require: Step size ϵ decay rate ρ
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2
Require: Small constant δ used for numeriacal stabilization
Require: Initial parameter $\boldsymbol{\theta}$
Initialise 1st and 2nd moments variables $\mathbf{s} = 0$, $\mathbf{r} = 0$
Initialise time step t=0
while stopping criterion not met **do**
  Sample a mini batch of m samples from the training set $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
  Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$.
  $t \leftarrow t + 1$
  Update first moment estimate: $\mathbf{s} \leftarrow \rho_1\mathbf{s} + (1-\rho_1)\mathbf{g}$.
  Update second moment estimate: $\mathbf{r} \leftarrow \rho_2\mathbf{r} + (1-\rho_2)\mathbf{g} \odot \mathbf{g}$
  Correct bias in the first moment: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1-\rho_1^t}$
  Correct bias in the second moment: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1-\rho_2^t}$
  Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon\hat{\mathbf{s}}}{\sqrt{\delta+\hat{\mathbf{r}}}}$.
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

4. Cosine annealing learning rate scheduler

Although Adam is generally regarded as being fairly robust to the choice of hyperparameters (Goodfellow et al.). The learning performance can be still improved from a good learning rate to an optimal learning rate. In section 4 we would explore how to further tune the learning rate and how these further tuned learning rates affects the learning performance.Zhang et al.showed the SGD with a tuned learning rate annealing schedule and momentum parameter achieves the competitive performance with Adam, and even converges faster. This work motivates us to apply an annealing learning rate scheduler in the Adam to explore whether the origin learning performance from Adam could be further improved. In addition, Loshchilov & Hutter proposed SGDR to improve performance of SGD by quickly cooling down the learning rate and periodically increasing it. In detailed, the learning rate would decrease rapidly according to the cosine annealing, then immediately restart. After restarting the optimization procedure continues from last step where parameters converge. To be more specific, the new high learning rate makes the parameters escape from the local minimum to a new loss surface. Then the cosine annealing would make the model rapidly converges to a new and potentially better optimal point. The Adam with a cosine annealing and warm restart is present in in algorithm 3. In out experiments we applied the restart scheme to the cosine annealing learning rate, then compared with one without restart to explore how the restart process affects the final performance. We set the initial setting of hyperparameters as ones in the provided test.py. The early stopping scheme also is implemented in this experiment. Figure 3 shows the trends of accuracy rate curve changing with respect to the epoch for SGD, SGD with cosine annealing and SGDR with cosine annealing and warm restart. Figure 3 also includes the accuracy rate curve for comparison of Adam, Adam with cosine annealing and Adam with cosine annealing with warm restart. As can be seen from the figure 3, for SGD there are not significant differences between using these learning schemes or without these learning schemes. Only in the validation set we could find SGD with cosine annealing and SGD with cosine annealing with warm restart have a slight improvement compared to SGD without such learning scheme. A noticeable point is in the case of SGD with cosine annealing with warm restart the accuracy rate curve has an area showing the clear trend of going down first then climbing up rapidly. This trend illustrates the effect of a restart to the learning procedure: the relatively high learning rate after restart makes the model escape from the local optimal point, which lowers the accuracy. Then as the learning procedure goes the tuned learning rate starts to help model look for a new optimal solution. Such a trend from restart are more clearly presented in the Adam. In the training set for Adam, there are not apparent differences between Adam with a cosine annealing and Adam without a cosine annealing. However the accuracy curve of Adam with cosine with restart shows a clear restart procedure. In the test set the Adam with cosine annealing performs better than Adam without cosine annealing. Adam with cosine annealing and warm restart achieves a highest accuracy rate. In early training the rapidly decreased learning rate (period is 25) might have an advantage effect to help the model find some local optimal point compared with Adam with only cosine annealing (period is 101). In addition during the procedure of tuning hyperparameter I noticed that the change of η_{max} has the most impact on the learning performance. So I set a search grid to look for a optimal η_{max}. A optimal hyperparameter η_{max} that I explored for Adam with cosine annealing and restart is 0.0009. Finally, the accuracy rates on the test set is following: SGD: 82.75%,SGD with cosine annealing: 82.76%, SGD with cosine annealing and warm restart: 82.81%; Adam: 82.37%, ,Adam with cosine annealing: 82.61%, Adam with cosine annealing and warm

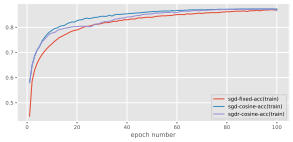
restart: 83.22%.

Algorithm 3 Adam with cosine annealing and warm restart

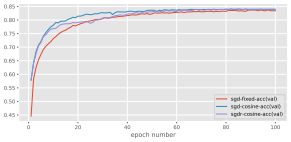
Require: Step size ϵ decay rate ρ
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2
Require: Small constant δ used for numeriacal stabilization
Require: Initial parameter $\boldsymbol{\theta}$
Initialise 1st and 2nd moments variables $\mathbf{s} = 0$, $\mathbf{r} = 0$
Initialise time step t=0
while stopping criterion not met **do**
  Sample a mini batch of m samples from the training set $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
  Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$.
  $t \leftarrow t + 1$
  Update first moment estimate: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$.
  Update second moment estimate: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1-\rho_2)\mathbf{g} \odot \mathbf{g}$
  Correct bias in the first moment: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1-\rho_1^t}$
  Correct bias in the second moment:$\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1-\rho_2^t}$
  $\eta \leftarrow$ SetScheduleMultiplier(t) (can be fixed, decay, or also be used for warm restart)
  Compute parameter update: $\Delta\boldsymbol{\theta} = -\eta \frac{\epsilon \hat{\mathbf{s}}}{\sqrt{\delta + \hat{\mathbf{r}}}}$.
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

5. Regularization and weight decay with Adam

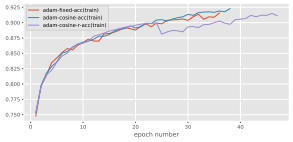
Both L2 regularization and weight decay are common and effective regularization. However, Loshchilov and Hutter pointed out there is the difference when they are applied in Adam, and Adam with decay generalizes much better than Adam with standard L2 regularization (Loshchilov et al. (2017)). In theory, L2 regularization accounts for adding an item $w\theta_{t-1}$ to the gradient, where w is the regularization hyperparameter. We can track the updated gradient step by step in Adam (Algorithm 2) until the step of parameter update. Then we can find actually θ_t will decay by $\epsilon w\theta_{t-1}$ rather than $w\theta_{t-1}$ as expected. Further if we want to keep the fixed decay factor $w\epsilon$ the regularization factor w should change as learning rate ϵ changes. Hence, these two hyperparameter would be dependent, which does not match our expectation. Such a dependent pair of hyperparameter would also be more difficult to tune, which imposes a negative effect to the final performance. In addition, from the parameter update step we can see the item $w\theta_{t-1}$from L2 regularization is also normalized, even worse, it is normalized by the summed amplitudes of both gradients of regularizer and loss function which leads a weaker decay especially for weights with large gradients. To solve the problem from the L2 regularization Loshchilov and Hutter directly add the weight decay item to the parameter update step. Such a simple modification simply makes the weight decay parameter w and learning rate ϵ independent. In the same time as the updated gradient does not include weight decay item, all the weights can be normalized by the same factor. Hence Adam with weight decay should have a better performance than the one with L2 regularization. One of our experiments in this section is designed to prove this theory by the comparison of Adam with L2 regularization and with weight decay. Beside this comparison it is also worth further exploring how cosine annealing learning rate and restart scheme in the weight decay affects the learning performance. As weight decay is a better strategy to decay the weight in Adam then will the application of cosine annealing and restart in AdamW provide a better performance than only using AdamW? Our experiments in this task will also cover this question. As before we initially set the hyperparameter as ones providing by coursework. We additionally set the hyperparameter for L2 regularizer as 0.00001 which keeps the same as the one for weight dacay. Early stopping is also included. Figure 4 shows the trends of accuracy rate curve changing with respect to the epoch for AdamW, AdamW with cosine annealing and AdamW with cosine annealing. Figure 4 also includes a comparison of Adam with L2 regularization and with weight decay. As can be seen from figure 4, at first for both training set and validation set we can clearly see the adam with weight decay performs better than Adam with L2 regularization. Especially in the validation set, after the early training the accuracy from AdamW almost keeps lead to Adam with L2 regularization, which proves the theory from Loshchilov and Hutter that AdamW generalizes much better than Adam with standard L2 regularization. For part of AdamW with cosine annealing and restart, AdamW with cosine annealing always perform better than AdamW without cosine annealing on both training set and validation set. AdamW with cosine annealing also perform better than AdamW with cosine annealing and restart in the training set while AdamW with cosine annealing and restart achieves the highest accuracy rate in the validation set. It suggests a short period (period is 25) cosine annealing in early training can help model rapidly converge to a local optimal solution while a



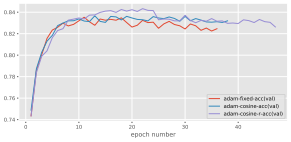
(a) Accuracy rate curve on training set for sgd with cosine annealing learning scheduler



(b) Accuracy rate curve on validation set for sgd with cosine annealing learning scheduler

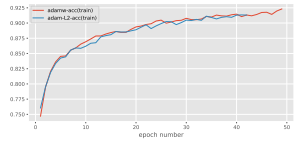


(c) Accuracy rate curve on training set for Adam with cosine annealing learning scheduler

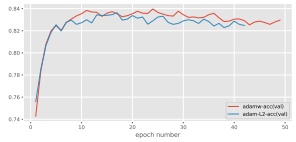


(d) Accuracy rate curve on validation set for Adam with cosine annealing learning scheduler

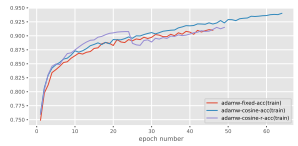
Figure 3: Accuracy rate curves for Adam and SGD with cosine annealing and warm restart



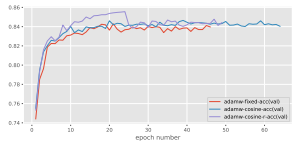
(a) Accuracy rate curve on training set for Adam with L2 regularization and AdamW



(b) Accuracy rate curve on validation set for for Adam with L2 regularization and AdamW



(c) Accuracy rate curve on training set for AdamW with cosine annealing learning scheduler



(d) Accuracy rate curve on validation set for AdamW with cosine annealing learning scheduler

Figure 4: Accuracy rate curves for Adam with L2 regularization, Adam with weight decay and AdamW with cosine annealing and warm restart

long period (period is 101) cosine annealing cannot provide great help for the model early convergence as the long period cosine function decreases relatively slowly. As Adam with cosine annealing and restart, the accuracy rate curve of AdamW with cosine annealing and restart also presents an remarkable effect from restart. During the period of training I also notice the η_{min} also has a relatively slight effect to the learning performance. So after setting an optimal η_{max}, I also implemented a grid search for η_{min}. The optimal setting of hyperparameter for AdamW with cosine annealing and warm restart is $\eta_{max} = 1.3$, $\eta_{min} = 0.0003$, and discount factor is 0.8. Finally the accuracy rate on the test set is following: Adam with L2 regularization: 82.48%,Adam with weight decay: 83.02%, AdamW: 82.78%, ,AdamW with cosine annealing: 82.99%, AdamW with cosine annealing and warm restart: 84.22%.

6. Conclusions

Based on my experiments, AdamW with cosine annealing and restart performs best on the Balanced EMNIST data, the final accuracy rate on the test set is 84.22%.

In these experiments we first implemented and explored two adaptive learning rate algorithms in the multi-layer neural network. Then basis on these two optimization algorithms, we also explored some current methods of adaptively tuning learning rates such as cosine annealing and warm restart. Many experiments has already proved a well-tuned learning rate is crucial to final learning performance.

In my experiments I found in some cases after restarting the accuracy rate is not improved to an ideal value that is better the optimal one before restart. So some future work might be done to explore how to implement a restart scheme more effectively. In addition I also notice weight decay always has a great help to the learning procedure. Hence, I also have interest to further explore the effect of the normalized weight decay to the final learning performance(Loshchilov et al.).

References

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. The MIT Press, 2017.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Loshchilov, Ilya, Hutter, and Frank, test. Fixing weight decay regularization in Adam. *arXiv preprint arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

Loshchilov, Ilya and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts. *In Proceedings of ICLR*, 2017.

Zhang, J., Mitliagkas, and Ré, C. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.