Journal of Machine Learning Research 24 (2023) 1-37 Submitted 5/21; Revised 2/23; Published 2/23

Wide-minima Density Hypothesis and the Explore-Exploit Learning Rate Schedule

Nikhil Iyer *
Microsoft Research India IYERNIKHIL007@GMAIL.COM

V Thejas *
Atlassian India THEJASVENKATESH97@GMAIL.COM

Nipun Kwatra
Microsoft Research India NKWATRA@MICROSOFT.COM

Ramachandran Ramjee
Microsoft Research India RAMJEE@MICROSOFT.COM

Muthian Sivathanu
Microsoft Research India MUTHIAN@MICROSOFT.COM

Editor: Honglak Lee

Abstract

Several papers argue that wide minima generalize better than narrow minima. In this paper, through detailed experiments that not only corroborate the generalization properties of wide minima, we also provide empirical evidence for a new hypothesis that the density of wide minima is likely lower than the density of narrow minima. Further, motivated by this hypothesis, we design a novel explore-exploit learning rate schedule. On a variety of image and natural language datasets, compared to their original hand-tuned learning rate baselines, we show that our explore-exploit schedule can result in either up to 0.84% higher absolute accuracy using the original training budget or up to 57% reduced training time while achieving the original reported accuracy.

Keywords: deep learning, generalization, learning rate schedule, optimization

1. Introduction

One of the fascinating properties of deep neural networks (DNNs) is their ability to generalize well, i.e., deliver high accuracy on the unseen test dataset. It is well-known that the learning rate schedules play an important role in the generalization performance (Keskar et al., 2016; Wu et al., 2018; Goyal et al., 2017). In this paper, we study the question, *what are the key properties of a learning rate schedule that help DNNs generalize well during training?*

We start with a series of experiments training Resnet18 on Cifar-10 over 200 epochs. We vary the number of epochs trained at a high learning rate of 0.1, called the *explore* epochs, from 0 to 100 and divide up the remaining epochs equally for training with learning rates of 0.01 and 0.001. Note that the training loss typically stagnates around 50 epochs with 0.1 learning rate. Despite that, we find that as the number of explore epochs increase to 100,

∗. Work done during an internship at Microsoft Research India

©2023 Nikhil Iyer, V.Thejas, Nipun Kwatra, Ramachandran Ramjee and Muthian Sivathanu.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v24/21-0549.html.

the average test accuracy also increases. We also find that the minima found in higher test accuracy runs are wider than the minima from lower test accuracy runs, corroborating past work on wide-minima and generalization (Keskar et al., 2016; Hochreiter and Schmidhuber, 1997; Jastrzebski et al., 2017; Wang et al., 2018). The width of a minimum is loosely defined as a measure of how slowly the loss value increases in the neighborhood of the minimum. We define the concrete measures used in this paper in section 3.2. Another observation that was particularly surprising was that, even when using fewer explore epochs, a few runs out of many trials still resulted in high test accuracies!

Thus, we not only find that an initial exploration phase with a high learning rate is essential to the good generalization of DNNs, but that *this exploration phase needs to be run for sufficient time, even if the training loss stagnates much earlier. Further, we find that, even when the exploration phase is not given sufficient time, a few runs still see high test accuracy values.*

To explain these observations, we hypothesize that, *in the DNN loss landscape, the density of narrow minima is significantly higher than that of wide minima.* Intuitively, a large learning rate can escape narrow minima easily (as the optimizer can jump out of them with large steps). However, once it reaches a wide enough minima, it is likely to get stuck in it since the escape time for SGD from a valley depends exponentially on the eigenvalue of the Hessian at the minima (Xie et al., 2020). With fewer explore epochs, a large learning rate might still get lucky occasionally in finding a wide minima but with high probability finds only a narrower minima due to their higher density. As the explore duration increases, the probability of eventually landing in a wide minima also increases. Thus, *a minimum duration of explore* is necessary to land in a wide minimum with high probability.

An observation on the rarity of wide minima has been hinted at by prior work (Wu et al., 2018; Baldassi et al., 2020) based on theoretical analysis of simple neural networks (see Section 2). In this paper, we add significant empirical evidence to these theoretical observations. We believe that all these results together constitute sufficient evidence for this observation to now be classified as a hypothesis, that we term the wide-minima density hypothesis.

The hypothesis helps *explain* not only our experiments but also the generalization out-performance of prior heuristic-based learning rate decay schemes such as cosine decay (Loshchilov and Hutter, 2016). Cosine decay *implicitly* maintains a higher learning rate during the first half of training compared to schemes like linear decay. Based on the hypothesis, the higher learning rate allows cosine decay to find wider minima with higher probability, resulting in cosine decay's better generalization compared to linear decay.

Apart from helping explain empirical observations, the hypothesis also enables a principled learning rate schedule design that *explicitly* accounts for the requisite explore duration. Motivated by the hypothesis, we design a novel *Explore-Exploit* learning rate schedule, where the initial *explore* phase optimizes at a high learning rate in order to arrive in the vicinity of a wide minimum. This is followed by an *exploit* phase which descends to the bottom of this wide minimum. We give *explore* phase enough time so that the probability of landing in a wide minima is high. For the *exploit* phase, we experimented with multiple schemes, and found a simple, parameter-less, linear decay to zero to be effective. *Thus, our proposed learning rate schedule optimizes at a constant high learning rate for a given duration, followed by a linear decay to zero. We call this learning rate schedule the Knee schedule.*

We extensively evaluate the *Knee schedule* across a wide range of models and datasets, ranging from NLP (BERT pre-training, Transformer on WMT'14(EN-DE) and IWSLT'14 (DE-EN)) to CNNs (ImageNet on ResNet-50, Cifar-10 on ResNet18), and spanning multiple optimizers: SGD Momentum, Adam, RAdam, and LAMB. In all cases, *Knee schedule* improves the test accuracy of state-of-the-art hand-tuned learning rate schedules, when trained using the original training budget. The explore duration is a hyper-parameter in *Knee schedule* but even if we set the explore duration to a fixed 50% fraction of total training budget, we find that it still outperforms prior schemes.

We also experimented with reducing the training budget, and found that *Knee schedule* can achieve the same accuracy as the baseline under significantly reduced training budgets. For the $\text{BERT}_{\text{LARGE}}$ pretraining, WMT'14(EN-DE) and ImageNet experiments, we are able to train in 33%, 57% and 44% less training budget, respectively, for the same test accuracy. This corresponds to significant savings in GPU compute, e.g. *savings of over 1000 V100 GPU-hours* for $\text{BERT}_{\text{LARGE}}$ pretraining.

The main contributions of our work [1] are:

1. A hypothesis of lower density of wide minima in the DNN loss landscape, backed by extensive experiments, that explains why a high learning rate needs to be *maintained for sufficient duration* to achieve good generalization.
2. The hypothesis *explains* the good performance of heuristic-based schemes such as cosine decay, and promotes a *principled design* of learning rate decay schemes.
3. Motivated by the hypothesis, we design an *Explore-Exploit* learning rate schedule called *Knee schedule* that outperforms prior heuristic-based learning rate schedules, including achieving state-of-the-art results on the IWSLT'14 (DE-EN) dataset.

2. Related Work

Generalization. There has been a lot of work on understanding the generalization characteristics of DNNs. Kawaguchi (2016) proved that for linear neural networks, the loss landscape can have many local minima, but all local minima are also the global minimum. It has been observed by several authors that wide minima generalize better than narrow minima (Arora et al., 2018; Hochreiter and Schmidhuber, 1997; Keskar et al., 2016; Jastrzebski et al., 2017; Wang et al., 2018) but there have been other works questioning this hypothesis as well (Dinh et al., 2017; Golatkar et al., 2019; Guiroy et al., 2019; Jastrzebski et al., 2019; Yoshida and Miyato, 2017).

Keskar et al. (2016) found that small batch SGD generalizes better and lands in wider minima than large batch SGD. However, recent work has been able to generalize quite well even with very large batch sizes (Goyal et al., 2017; McCandlish et al., 2018; Shallue et al., 2018), by scaling the learning rate linearly as a function of the batch size. Jastrzebski et al. (2019) analyze how batch size and learning rate influence the curvature of not only the SGD endpoint but also the whole trajectory. They found that small batch or large step SGD have similar characteristics, and yield smaller and earlier peak of spectral norm as well as smaller largest eigenvalue. Chaudhari et al. (2019); Baldassi et al. (2020) propose methods to drive the optimizer to wide minima. Wang et al. (2018) analytically show that generalization of a model is related to the Hessian and propose a new metric for the

1. Source code available at: `https://github.com/nikhil-iyer-97/wide-minima-density-hypothesis`

generalization capability of a model that is unaffected by model reparameterization of Dinh et al. (2017). Yoshida and Miyato (2017) argue that regularizing the spectral norm of the weights of the neural network help them generalize better. On the other hand, Arora et al. (2018) derive generalization bounds by showing that networks with low stable rank (high spectral norm) generalize better. Guiroy et al. (2019) looks at generalization in gradient-based meta-learning and they show experimentally that generalization and wide minima are not always correlated. Finally, Golatkar et al. (2019) show that regularization results in higher test accuracy specifically when it is applied during initial phase of training, similar to the importance of *Knee schedule*'s explore phase during initial phase of training. In a similar vein, Li et al. (2019) explain the regularization benefits of the initial higher learning rate by showing that higher learning rate helps networks learn easier-to-fit general patterns. Li et al. (2020) show that for scale invariant networks, the effective speed of learning is inversely proportional to the "intrinsic learning rate" (product of learning rate and weight-decay), and that one can achieve high test accuracy even with low learning rate. Yue et al. (2020) proposes an adaptive learning rate schedule, SALR, where the learning rate is increased if the local landscape is sharp. This allows them to escape sharp minima, similar to our method. They show gains over other methods aimed at improving generalization, such as Stochastic Weight Averaging (Izmailov et al., 2018) and Entropy-SGD (Chaudhari et al., 2019). SALR, however, requires more computation per step as it needs to compute local sharpness. Also they achieve lower accuracy compared to our *Knee schedule*. For example, they report an accuracy of 94.94 with ResNet50 on Cifar10, while we obtain an accuracy of 95.26 with Resnet18 (a much smaller network) for the same number of gradient computation steps.

Neural network loss landscapes. The landscape of loss in neural networks have been extensively studied (Draxler et al., 2018; Freeman and Bruna, 2016; Garipov et al., 2018; Sagun et al., 2017). These papers point out that the loss landscape contains both wide and narrow minima, and there may even exist a path from one minima to another without barriers. However, there are multiple paths between these minima and some paths indeed face barriers (e.g., see Figure 1 in Draxler et al. (2018)). Since we don't know which path SGD and other optimizers might follow, even if wide and narrow minima are part of a single basin, SGD and other optimizers might still require higher learning rates to navigate from narrow to wide minima. Recent work has also shown interesting results for loss landscape trajectory followed by SGD for networks trained to minimize L2 loss with added label noise (Blanc et al., 2020; Damian et al., 2021; Li et al., 2021). They find that SGD first tracks full batch gradient descent until it gets close to the manifold of zero training error. In the second phase it optimizes an implicit training loss within this manifold of zero training error, which drives the solution to only "simple" models. Although their setting (L2 loss and label noise) is different from that explored in this paper, it suggests an alternative to our hypothesis and needs further analysis.

Lower density of wide minima. Wu et al. (2018) compares the sharpness of minima obtained by full-batch gradient descent (GD) with different learning rates for small neural networks on FashionMNIST and Cifar10 datasets. They find that GD with a given learning rate finds the theoretically sharpest feasible minima for that learning rate. Thus, in the presence of several flatter minimas, GD with lower learning rates does not find them, leading to the conjecture that density of sharper minima is perhaps larger than density of wider

minima. Baldassi et al. (2020) show analytically for simple, two-layer non-convex networks that wide minima exists and are rare, compared to narrow minima, local minima and saddle points. In this paper, we add significant evidence to these theoretical observations based on empirical results obtained on large-scale, state-of-the-art neural networks through carefully designed experiments.

3. Wide-Minima Density Hypothesis

Many popular learning rate schedules, such as the step decay schedules for image datasets, start the training with high learning rate, and then reduce the learning rate periodically. For example, consider the case of Cifar-10 on Resnet-18, trained using a typical step learning rate schedule of $0.1, 0.01$, and 0.001 for 100, 50, 50 epochs each. In many such schedules, even though training loss stagnates after several epochs of high learning rate, one still needs to continue training at high learning rate in order to get good generalization.

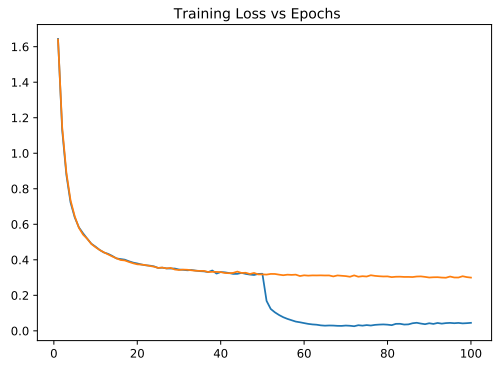


Figure 1: Training loss for Cifar-10 on Resnet-18. Orange plot uses a fixed learning rate of 0.1, while in blue plot, the learning rate is reduced from 0.1 to 0.01 at epoch 50.

Table 1: Cifar-10 on Resnet-18 trained for 200 epochs with Momentum. A learning rate of 0.1 is used for the explore epochs. Half the remaining epochs are trained at 0.01 and the other half at 0.001. Reported results are average over 4 runs.

Epochs at 0.1 LR	Test Accuracy Avg. (Std. Dev)	Train Loss Avg. (Std. Dev.)
0	94.34 (0.13)	0.0017 (8e-5)
30	94.81 (0.15)	0.0017 (8e-5)
40	94.91 (0.14)	0.0018 (9e-5)
60	95.01 (0.14)	0.0018 (1e-4)
80	95.05 (0.15)	0.0019 (1e-4)
100	95.10 (0.14)	0.0021 (1e-4)

For example, Figure 1 shows the training loss for Cifar-10 on Resnet-18, trained with a fixed learning rate of 0.1 (orange curve), compared to a model trained via a step schedule with learning rate reduced at epoch 50 (blue curve). As can be seen from the figure, the training loss stagnates after ≈ 50 epochs for the orange curve, and locally it makes sense to reduce the learning rate to decrease the loss. However, as shown in Table 1, generalization is directly correlated with duration of training at high learning rate, with the highest test accuracy achieved when the high learning rate is used for 100 epochs, well past the point where training loss stagnates. Note that the final training loss remains similar for all runs.

To understand the above phenomena, we perform another experiment. We train Cifar-10 on Resnet-18 for 200 epochs, using a high learning rate of 0.1 for only 30 epochs and then use learning rate of 0.01 and 0.001 for 85 epochs each. We repeat this training 50 times with different random weight initializations. On an average, as expected, this training yields a low test accuracy of 94.81. However, *in 1 of the 50 runs, we find that the test accuracy*

reaches 95.24*, even higher than the average accuracy of* 95.1 *obtained while training at high learning rate for 100 epochs!*

3.1 Hypothesis

To explain the above observations, i.e., using a high learning rate for *short duration* results in *low average test accuracy with rare occurrences of high test accuracy*, while using the same high learning rate for *long duration* achieves *high average test accuracy and frequent occurrences of high test accuracy*, we introduce a new hypothesis. We hypothesize that, *in the DNN loss landscape, the density of narrow minima is significantly higher than that of wide minima.* To define the hypothesis more formally, we first define a few terms.

Definition 1 (Basin of Attraction) *The basin of attraction $\mathbb{B}(m)$ of a minimum m is defined as the set of points in the parameter space, such that when a gradient-flow algorithm is initialized from a point $p \in \mathbb{B}(m)$, the algorithm converges to the minimum m.*

One can also define a *relaxed* version of the above definition, in which gradient-flow is replaced with gradient-descent using a small time-step δ. This results in smoothing out small perturbations (of magnitude δ) in the loss landscape, when defining the basins.

Remark 1.1 *Almost all points in the parameter space belong to some basin of attraction, $\mathbb{B}(m)$. This implies that the parameter space can be partitioned into a set of basins corresponding to the different minima.*

This follows from the convergence properties of gradient-flow (see Swenson et al. (2020)), where under certain assumptions (on smoothness and bounds) on the loss function, we can show that gradient flow converges to a unique local minimum or to a saddle point. The set of initialization points which cause gradient-flow to converge to a saddle point is a set (called *stable manifold*) of measure zero. Thus, almost all (ignoring the small stable manifold set) points in the parameter space can be partitioned into different basins.

Definition 2 (Wide and Narrow Minima) *Given a minima sharpness metric (see Section 3.2), and a suitably chosen threshold[2] s, we define a minimum as wide if its sharpness is less than s, and narrow otherwise.*

Definition 3 (Minima Density) *Given a point p in the parameter space, construct a ball $S(p)$ around the point p, such that it is large enough to contain k basins. Here k is some chosen large number. Let k_w out of the k basins correspond to wide minima and k_n correspond to narrow minima. We then define the density of wide minima in the neighborhood of p as k_w/V, and the density of narrow minima as k_n/V, where V is the volume of the ball $S(p)$.*

Our hypothesis can then be more formally stated as:

Hypothesis 1 *Consider any point p in the parameter space of a DNN. The density of narrow minima in the neighborhood of p is significantly higher than that of wide minima.*

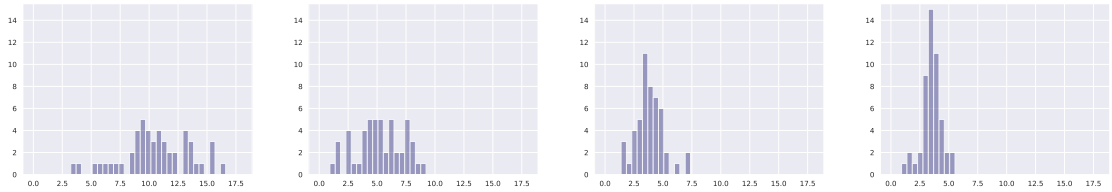
2. The threshold can be chosen empirically based on the relationship between sharpness and test accuracy (see Table 2 for example).

The hypothesis can explain our observations as follows. Intuitively, a large learning rate can escape narrow minima basins easily (as the optimizer can jump out of them with large steps). However, once it reaches a wide minima basin, it is likely to get stuck in it (if the "width" of the wide basin is large compared to the step size). Although noise in stochastic optimizers such as SGD may allow escape from a wide minima, the probability of such escape will be inversely related to the minimum width. This intuition is backed by theoretical results from Xie et al. (2020) that show that the *time to escape a minimum using SGD is exponential in the inverse of learning rate as well as inverse of the sharpness* (measured by eigenvalue of the Hessian at the minima). Thus, large learning rates escape narrow minima exponentially faster than wide minima.

If wide and narrow minima were uniformly distributed, SGD with a large LR would be able to quickly escape the narrow minima, land on a wide minima and get stuck there. Yet, we see that we need to maintain large LR for significant duration for landing in a wide minima with high probability. On the other hand, if our hypothesis is true, i.e., wide minima are much fewer than narrow minima, the probability of landing in a wide minima after escaping a narrow minima is low, and the optimizer needs to take a lot of steps to have a high probability of eventually landing in a wide minimum. Thus, the hypothesis is a better explanation for the observation in Table 1, where the average accuracy continues to improve as we increase the number of high learning rate training steps. The hypothesis also explains why very few (just 1) of the 50 runs trained at 0.1 learning rate for just 30-epochs also manages to attain high accuracy—these runs just got lucky in a probabilistic sense and landed in a wide minimum even with a shorter duration of explore.

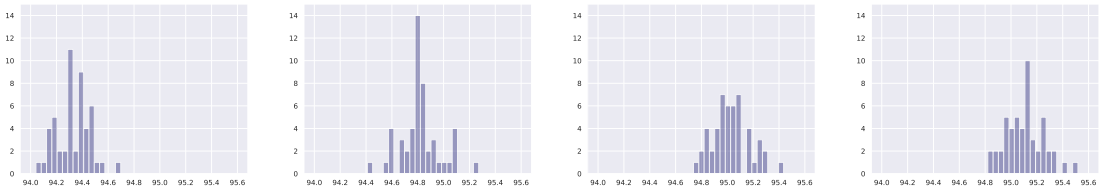
To validate this hypothesis further, we run experiments similar to the one in Table 1. Specifically, we train Cifar-10 on Resnet-18 model for 200 epochs using a standard step schedule with learning rate of $0.1, 0.01, 0.001$. We vary the number of epochs trained using the high learning rate of 0.1, called the *explore epochs*, from 0 to 100 epochs, and divide up the rest of the training equally between 0.01 and 0.001. For each experimental setting, we conduct 50 random trials and plot the distributions of final test accuracy and the minima sharpness as defined by the metric in Keskar et al. (2016) (see section 3.2). If our hypothesis is true, then the more you explore, the higher the probability of landing (and getting stuck) in a wide minima region, which should cause the distribution to tighten and move towards wider minima (lower sharpness), as the number of explore steps increase. This is exactly what is observed in Figure 2. Also since wide minima correlate with higher test accuracy, we should see the test accuracy distribution move towards higher accuracy and sharpen, as the number of explore steps increase. This is confirmed as well in Figure 3.

Longer training with low learning rate is not sufficient. Finally, to verify whether explore at high learning rate is essential, we train Cifar-10 for 10,000 epochs at a fixed lower learning rate of 0.001. The training loss converged but the final test accuracy was only **93.9**, compared to an accuracy of over 95% in 200 epochs in Table 1. Thus, even training $50\times$ longer at low learning rate is not sufficient to achieve good generalization. Again, this observation ties in well with the theoretical results from Xie et al. (2020) where the authors show that the time to escape a minimum using SGD is exponential in the inverse of learning



(a) Explore 0 (b) Explore 30 (c) Explore 60 (d) Explore 100

Figure 2: Histogram of minima sharpness (Keskar et al., 2016) for 50 random trials of Cifar-10 on Resnet-18. Each figure shows histograms for runs with different number of explore epochs. The distribution moves toward lower sharpness and tightens as the number of explore epochs increase.



(a) Explore 0 (b) Explore 30 (c) Explore 60 (d) Explore 100

Figure 3: Histogram of test accuracy for 50 random trials of Cifar-10 on Resnet-18. Each figure shows histograms for runs with different number of explore epochs. The distribution moves toward higher test accuracy and sharpens as the number of explore epochs increase.

rate. Thus, *this result adds further evidence to our density hypothesis, since even training* $50\times$ *longer at a low learning rate is not sufficient to land in a wide minima.*[3]

Multi-scale. Given the importance of explore at high learning rate, a natural question that may arise is whether explore is necessary at smaller learning rate as well. To answer this, we train the same network for a total of 200 epochs with an initial high learning rate of 0.1 for 100 epochs, but now we vary the number of epochs trained with the learning rate of 0.01 (we call this finer-scale explore), and train with learning rate of 0.001 for the remaining epochs. As can be seen from Table 2, although the final training loss remains similar, we find that finer-scale explore also plays a role similar to the initial explore in determining the final test accuracy. *This indicates that our hypothesis about density of wide/narrow regions indeed holds at multiple scales.*

3.2 Minima Sharpness

Our hypothesis predicts that higher explore helps the optimizer land in a wider minimum, which in turn helps generalization. We demonstrated this empirically in Figure 2, where we plotted the distribution of the minima sharpness, as measured by the sharpness metric introduced by (Keskar et al., 2016). In this section, we describe Keskar's sharpness metric in

3. Note that for scale-invariant networks with batch normalization, Li et al. (2020) show that one can achieve high test accuracy with SGD, when trained with a low learning rate for a very large number of epochs. However, it is not clear if the same holds for non scale-invariant networks such as the ones used in this paper.

Table 2: Cifar-10 on Resnet-18 trained for 200 epochs. A learning rate of 0.1 is used for the first 100 epochs. We then vary the number of epochs trained with learning rate of 0.01 (called finer-scale explore), and train the remaining epochs with a learning rate of 0.001. We report averages values over 3 runs.

Explore Epochs (Finer-scale)	Test Accuracy	Training Loss	Sharpness
10	94.78	0.0031	5.48
20	94.91	0.0026	4.47
30	95.00	0.0023	4.02
40	95.02	0.0021	3.91
50	95.10	0.0021	3.54

detail. We also introduce a simple projected gradient ascent scheme to compute this metric efficiently, which scales well to large networks. Finally, we also evaluate our hypothesis with a different metric for minima sharpness, the *Fisher Score*, which is based on the Fisher information matrix.

3.2.1 Keskar's Sharpness Metric

Keskar's sharpness metric is based on measuring the maximum jump in the network's output function F in a small neighborhood around the minimum. After a few simplifications, Keskar's metric for sharpness around a point x can be written as:

$$S_{x,F}(\epsilon) := \frac{(max_{y \in C_\epsilon(x)} F(x+y)) - F(x)}{1 + F(x)} \times 100, \tag{1}$$

where $C_\epsilon(x)$ is an ϵ neighborhood around x. Keskar et al. (2016) mentions that under certain conditions and for small values of ϵ, $S_{x,F}$ is proportional to the largest eigenvalue of the Hessian. Please see Keskar et al. (2016) for more details. For our measurements we choose an ϵ of $1e^{-4}$.

For solving the maximization problem in Equation 1, Keskar et al. (2016) uses a second-order L-BFGS-B (Byrd et al., 2003) optimization scheme. However, in our experiments we found the method to be very slow. To combat this, Keskar et al. (2016) limited their runs to 10 iterations but we found that results were suboptimal using few iterations. Instead, we employed a projected gradient ascent scheme to solve Equation 1. In each optimization step, we took a small step with a learning rate of 0.001 in the gradient direction and projected the updated point to lie inside $C_\epsilon(x)$. Because of the first order nature, this method is much faster. We found that even 1000 iterations were fast to compute and the results were much better than the second order method in all cases we evaluated.

Using Keskar's sharpness metric, we had shown in Figure 2 that the distribution of minima sharpness moves towards lower values as the number of explore epochs increase. In Table 3, we also report the average sharpness of the minima for varying explores. As predicted by our hypothesis, average sharpness decreases as number of explore epochs increase.

Table 3: Keskar's sharpness metric for Cifar-10 on Resnet-18 trained for 200 epochs with Momentum. A learning rate of 0.1 is used for the explore epochs. Half the remaining epochs are trained at 0.01 and the other half at 0.001. We report the average sharpness over 50 different trials.

Explore Epochs	Sharpness
0	10.56
30	5.43
60	3.86
100	3.54

3.2.2 **Fisher Score**

The maximum Eigen value of the Fisher Information Matrix (FIM) estimates the highest curvature at a point, and is used as another metric to measure minima sharpness (Sokol and Park, 2018). We used an unbiased estimate of the true Fisher matrix (see Kunstner et al. (2019)) using 10 unbiased samples per training data. Table 4 shows the average Fisher scores for the Cifar-10 experiments at varying explores. Again, the sharpness measured by the Fisher score decreases as the number of explore epochs increase.

Table 4: Fisher Score for Cifar-10 on Resnet-18 trained for 200 epochs with Momentum. A learning rate of 0.1 is used for the explore epochs. Half the remaining epochs are trained at 0.01 and the other half at 0.001. We report the average Fisher score over 10 different trials.

Explore Epochs	FIM score
0	0.051
30	0.046
60	0.043
100	0.042

4. Explore-Exploit Learning Rate Schedule

Given that we need to explore at multiple scales for good generalization, how do we go about designing a good learning rate schedule? The search space of the varying learning rate steps and their respective explore duration is enormous.

Fortunately, since the explore at the initial scale is searching over the entire loss surface while explore at finer-scales is confined to exploring only the wide-minima region identified by the initial explore, the former is more crucial. In our experiments as well, we found that the initial portion of training is much more sensitive to exploration and needs a substantial number of *explore* steps, while after this initial phase, several decay schemes worked equally well. This is similar to the observations in (Golatkar et al., 2019) where the authors found that regularization such as weight-decay and data augmentation mattered significantly only during the initial phase of training.

The above observations motivate our *Explore-Exploit* learning rate schedule, where the *explore* phase first optimizes at a high learning rate for some minimum time in order to land in the vicinity of a wide minima. We should give the *explore* phase enough time (a hyper-parameter), so that the probability of landing in a wide minima is high. After the *explore* phase, we know with a high probability, that the optimizer is in the vicinity of a wide region. We now start the *exploit* phase to descend to the bottom of this wide region while progressively decreasing the learning rate. Any smoothly decaying learning rate schedule

can be thought of as doing micro *explore-exploit* at progressively reduced scales. A steady descent would allow more *explore* duration at all scales, while a fast descent would explore less at higher learning rates. We experimented with multiple schedules for the exploit phase, and found a simple linear decay to zero, that does not require any hyper-parameter, to be effective in all the models/datasets we tried. We call our proposed learning rate schedule which starts at a constant high learning rate for some minimum time, followed by a linear decay to zero, the *Knee schedule*.

Note that any learning rate decay scheme incorporates an implicit explore during the initial part, where the learning rate stays high enough. To evaluate the benefit of an explicit explore phase, we compare *Knee schedule* against several decay schemes such as linear and cosine. Interestingly, the results depend on the length of training. For long budget experiments, simple decay schemes perform comparable to *Knee schedule* in some experiments, since the implicit explore duration is also large, helping these schemes achieve good generalization. However for short budget experiments, these schemes perform significantly worse than *Knee schedule*, since the implicit explore duration is much shorter. See Table 5 , 6 and 7 for the comparison.

Warmup. Some optimizers such as Adam use an initial warmup phase to slowly increase the learning rate. However, as shown in Liu et al. (2019), learning rate warmup is needed mainly to reduce variance during initial training stages and can be eliminated with an optimizer such as RAdam. Learning rate warmup is also used for large-batch training (Goyal et al., 2017). Here, warmup is necessary since the learning rate is scaled to a very large value to compensate for the large batch size. This warmup is complementary and can be incorporated into *Knee schedule*. For example, we do this for $\text{BERT}_{\text{LARGE}}$ pretraining experiment where a large 16k batch size was used.

5. Evaluation

In this section we present extensive empirical evaluation of *Knee schedule* on multiple models and datasets across various optimizers, and compare *Knee schedule* against the original hand-tuned learning rate baselines. We first provide an overview of our main results followed by detailed experimental results. We then run further experiments to validate our wide-minima density hypothesis, as well as run sensitivity analysis of seed learning rate on the *Knee schedule*.

Note that, for completeness, we present a detailed comparison of *Knee schedule* with many other learning rate schedules in literature such as linear decay, cosine decay (Loshchilov and Hutter, 2016) and one-cycle (Smith, 2018) in Appendix A.

5.1 Experiments

We evaluate *Knee schedule* on multiple models and datasets spanning both vision and NLP problems. The training of these models spanned various optimizers including SGD Momentum, Adam (Kingma and Ba, 2014), RAdam (Liu et al., 2019) and LAMB (You et al., 2019). For all experiments, we used an out of the box policy, where we only change the learning rate schedule, without modifying anything else. We evaluate on multiple image datasets – Imagenet on Resnet-50, Cifar-10 on Resnet-18; as well as various NLP datasets

– pretraining $\text{BERT}_{\text{LARGE}}$ on Wikipidea+BooksCorpus and fine-tuning it on SQuADv1.1; and WMT'14 (EN-DE), IWSLT'14 (DE-EN) on Transformers.

5.2 Results Overview

In all our experiments, we find that *Knee schedule shows an improvement in test accuracy over the original hand-tuned learning rate baseline as well as various other learning rate schedules in the literature. Further, we also find that Knee schedule can achieve the same accuracy as the baseline with a much reduced training budget.*

Table 5: We report the top-1 accuracy for ImageNet and Cifar-10, BLEU score for IWSLT'14 and WMT'14 and F1 score for BERT on SQuAD. All values are averaged over multiple runs for each experiment. Experiment details are mentioned in the individual sections of the experiments.

Experiment	Training Budget (epochs)	*Knee Schedule*	*Knee Schedule* (Fixed 50% explore)	Baseline	One-Cycle Decay	Cosine Decay	Linear
ImageNet	90	**76.71**	76.58	75.87	75.39	76.41	76.54
Cifar-10	200	**95.26**	95.26	95.10	94.09	95.23	95.18
IWSLT	50	**35.53**	35.23	34.97	34.77	35.21	34.97
WMT'14	70	**27.53**	27.41	27.29	27.19	27.35	27.29
$\text{BERT}_{\text{LARGE}}$	31250 (iters)	**91.51**	91.51	91.34	-	-	91.34

Table 6: Shorter budget training: Test accuracy on all learning rate schedules tried in this paper, but trained with a shortened budget. We report same metrics as Table 5. *Knee schedule* achieves the same accuracy as baseline schedules using much lower budget, saving precious GPU-hours.

Experiment	Shortened Training Budget (epochs)	*Knee Schedule*	One-Cycle	Cosine Decay	Linear Decay	Saving (V100 GPU hours)
ImageNet	50	**75.92**	75.36	75.71	75.82	27
Cifar-10	150	**95.14**	93.84	95.06	95.02	0.25
IWSLT	35	**35.08**	34.43	34.46	34.16	0.75
WMT'14	30	**27.28**	26.80	26.95	26.77	80
$\text{BERT}_{\text{LARGE}}$	20854 (iters)	**91.29**	-	-	90.64	1002

Table 5 shows the test accuracies of the various experiments, when trained with the original budget; while Table 6 shows the results when trained with a reduced budget. As shown, for the original budget runs, *Knee schedule* improves on the test accuracies in all experiments. Note that in *Knee schedule*, the explore duration is a hyperparameter. To avoid tuning this hyperparameter, we experimented with a fixed 50% explore duration for the full budget runs. Even the fixed 50% explore *Knee schedule* outperforms all the other

Table 7: Epochs required by different LR schedules to reach the target accuracy. The target accuracy is chosen based on *Knee schedule*'s results with a reduced budget.

Experiment	Target BLEU Score	*Knee schedule*	Cosine Decay	Linear Decay
IWSLT	35.08	35	45	60
WMT'14	27.28	30	60	70

baselines. Also noteworthy is that *Knee schedule* is able to achieve the same test accuracies as the baseline's full budget runs with a much lower training budget, saving precious GPU cycles (Table 6).

While the difference in accuracy values between the various schedules might appear deceptively small in absolute terms, achieving these gains require a large amount of compute. For example, the number of epochs needed by each scheme to reach the target BLEU score for IWSLT'14 DE-EN and WMT'14 EN-DE with the Transformer network is shown in Table 7. One can see that *Knee schedule* is significantly more efficient as compared to say Cosine Decay, which takes 100% more training time to achieve the same accuracy for WMT'14 EN-DE. Thus, the accuracy and/or compute gains achieved by *Knee schedule* is significant.

A summary of our main experimental results is as follows:

1. Imagenet on Resnet-50: We show an absolute gain of 0.8% in top-1 accuracy against the competitive step schedule baseline for this model. Also, *Knee schedule* can achieve the same accuracy as baseline in ∼45% less training epochs.

2. $\text{BERT}_{\text{LARGE}}$ pre-training on Wikipedia+BooksCorpus dataset: Compared to the baseline of You et al. (2019), we improve the F1 score on SQuAD v1.1 fine-tuning task by 0.2% (91.51 compared to 91.34). Also, we were able to achieve similar accuracy as baseline in 33% less training steps (a saving of ∼1002 V100 GPU-hours!).

3. WMT'14 and IWSLT machine translation on Transformers: Compared to competitive baselines, we were able to improve the BLEU scores by 0.24 and 0.56 points for the two tasks. Moreover, *Knee schedule* was able to achieve the same accuracy as baselines in 57% and 30% less training times.

4. Results on high accuracy model: We also show results on the IWSLT'14(DE-EN) machine translation dataset by simply replacing the learning rate schedule of a high accuracy model (Shen et al., 2020) with *Knee*. We were able to improve the BLEU score by 0.18, reaching a score of 37.78. Moreover, *Knee* can achieve the baseline accuracy in 30% less training time.

5.3 Detailed Results

We now describe each of our main experimental results in detail.

5.3.1 ImageNet Image Classification on Resnet-50

We train ImageNet dataset (Russakovsky et al., 2015) on Resnet-50 network (He et al., 2016) which has 25 million parameters, with a batch size of 256 and a seed learning rate of 0.1. Random cropping and random horizontal flipping augmentations were applied to the training dataset. We use SGD optimizer with momentum of 0.9 and weight decay of $1e^{-4}$. For baseline runs, we used the standard hand-tuned step learning rate schedule of 0.1, 0.01 and 0.001 for 30 epochs each. For *Knee schedule* we used a seed learning rate of 0.1 (same as baseline). We trained with the original budget of 90 epochs as well as with a reduced budget of 50 epochs. We used 30 explore epochs for the two experiments. [4]

Table 8 shows the training loss and test accuracies for our experiments. *Knee schedule* comfortably beats the test accuracy of baseline in the full budget run (with absolute gains of 0.8% and 0.4% in top-1 and top-5 accuracy, respectively), while meeting the baseline accuracy even with a much shorter budget. The fact that the baseline schedule takes almost 80% more training time than *Knee schedule* for the same test accuracy, shows the effectiveness of our *Explore-Exploit* scheme. See Figure 5 in Appendix B for training curves.

Table 8: ImageNet on Resnet-50 results. We report mean (stddev) over 3 runs.

LR Schedule	Test Top 1 Acc.	Test Top 5 Acc.	Training Loss	Training Epochs
Baseline	75.87 (0.035)	92.90 (0.015)	0.74 (1e-3)	90
Knee	76.71 (0.097)	93.32 (0.031)	0.79 (1e-3)	90
Knee (short budget)	75.92 (0.11)	92.90 (0.085)	0.90 (3e-3)	50

5.3.2 Cifar-10 Image Classification on Resnet-18

We train Cifar-10 dataset (Krizhevsky et al., 2009) on Resnet-18 network (He et al., 2016) which has around 11 million parameters. SGD optimizer is used with momentum of 0.9 and weight decay of $5e^{-4}$. Random cropping and random horizontal flipping augmentations were applied to the training dataset. [5].

For baseline, we used the hand-tuned step learning rate schedule of 0.1, 0.01 and 0.001 for 100, 50, 50 epochs, respectively. With *Knee schedule*, we train the network with the original budget of 200 epochs, as well as a reduced budget of 150 epochs. We used 100 explore epochs for both runs, and a seed learning rate of 0.1 (same as baseline). Table 9 shows the training loss and test accuracies for the experiments. *Knee schedule* beats the test accuracy of baseline in the full budget run, while meeting the baseline test accuracy in 25% less budget. Refer to figure 6 in Appendix B for detailed comparisons of training loss, test accuracy, and learning rate.

5.3.3 $\text{BERT}_{\text{LARGE}}$ Pre-training

We pretrain on $\text{BERT}_{\text{LARGE}}$ on Wikipedia+BooksCorpus dataset with LAMB optimizer (You et al. (2019)). $\text{BERT}_{\text{LARGE}}$ has around 330 million parameters and the pre-training is divided into two phases with different sequence lengths. The first phase consists of 90%

4. We used the opensource implementation at: `https://github.com/cybertronai/imagenet18_old`
5. We used the open-source implementation at: `https://github.com/kuangliu/pytorch-cifar`

Table 9: Training loss and Test accuracy for Cifar-10 on Resnet-18. We report mean (stddev) over 7 runs.

LR Schedule	Test Accuracy	Training Loss	Training Epochs
Baseline	95.10 (0.14)	0.002 (1e-4)	200 epochs
Knee	95.26 (0.11)	0.002 (1e-4)	200 epochs
Knee (short budget)	95.14 (0.18)	0.004 (3e-4)	150 epochs

steps with sequence length of 128 and the second phase consists of the remaining 10% steps with sequence length of 512 (Devlin et al. (2018)). We used a batch size of 16384 in both phases of training [6]. We use the same training budget of 31250 steps mentioned in (You et al. (2019)). We also train the model on a shortened training budget of $2/3^{\text{rd}}$ the original steps (20854 steps).

Since large batch training requires learning rate warmup (see Goyal et al. (2017)), we incorporate it into the *Knee schedule* by first doing a warmup of 10% as suggested in (You et al., 2019) followed by the explore-exploit phases. We used an explore of 50% of the total steps available for both phases of BERT training. For baseline, we use the warmup (10%) + linear decay (90%) schedule (You et al., 2019; Devlin et al., 2018). The pre-trained models are evaluated on the SQuAD v1.1 (Rajpurkar et al., 2016) dataset by fine-tuning on the dataset for 2 epochs. See Table 10 for the results. For the full budget run, *Knee schedule* improves the baseline by 0.2%, while for the reduced budget we achieved similar fine-tuning accuracy as baseline. The baseline schedule achieves a much lower accuracy with shorter budget training, showing the efficacy of *Knee schedule*. BERT pre-training is extremely compute expensive and takes around 47 hours on 64 V100 GPUs (3008 V100 GPU-hrs) on cloud VMs. The reduced budget amounts to a saving of approximately 1002 V100 GPU-hours!

Table 10: $\text{BERT}_{\text{LARGE}}$ results. We report the pre-training train loss, and the test F1 accuracy on SQuAD v1.1 after fine-tuning. See figure 7 in Appendix B for training curves.

LR Schedule	F1 score on SQuAD v1.1	Training loss	Total Training Steps
Knee	91.51	1.248	31250
Baseline (You et al., 2019)	91.34	-	31250
Baseline (short budget)	90.64	1.336	20854
Knee (short budget)	91.29	1.275	20854

5.3.4 Machine Translation on Transformer Network with WMT'14 and IWSLT

In the second NLP task, we train the Transformer (base model) (Vaswani et al., 2017) on the IWSLT'14 (De-En) (Cettolo et al., 2014) and WMT'14 (En-De) (Bojar et al., 2014) datasets with the RAdam (Liu et al., 2019) optimizer.

6. We used the open-source implementation at:
`https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT`

WMT'14 (EN-DE): We use the default implementation provided by the fairseq package (Ott et al., 2019) [7]. We train WMT'14 (EN-DE) dataset on the Transformer$_{\text{BASE}}$ (Vaswani et al., 2017) model which has around 86 million parameters and use the RAdam (Liu et al., 2019) optimizer with β_1 of 0.9 and β_2 of 0.999. Label smoothed cross entropy was used as the objective function with an uncertainty of 0.1. A dropout of 0.1, clipping norm of 25 and weight decay of $1e^{-4}$ is used. Each training batch contains approximately 30000 tokens.

The baseline schedule uses a linear decay for 70 epochs (Liu et al., 2019). With *Knee schedule*, we trained with the original budget of 70 epochs, as well as a reduced budget of 30 epochs. We used 50 and 25 explore epochs for the two runs, respectively and a seed learning rate of $3e^{-4}$ for both *Knee schedule* and baseline. In all cases we use the model checkpoint with least loss on the validation set for computing BLEU scores on the test set. Table 11 shows the training loss and test accuracy averaged over 3 runs. *Knee schedule* improves the test BLEU score of baseline in the full budget run by 0.24 points. In the shorter budget run, *Knee schedule* matches the test accuracy of the baseline while taking 57% less training time (a saving of 80 V100 GPU-hours!). See Figure 8 in Appendix B for training curves.

IWSLT'14 (DE-EN): For IWSLT'14 (DE-EN) we use the same configuration as WMT'14 (EN-DE), except for a dropout of 0.3 following Fairseq's out-of-box implementation. Each training batch contains approximately 4000 tokens. For *Knee schedule*, we choose explore as 30 epochs for short budget runs and 40 epochs for full budget runs.

The baseline schedule uses a linear decay for 50 epochs (Liu et al., 2019). With *Knee schedule*, we trained with the original budget of 50 epochs, as well as a reduced budget of 35 epochs. We used 40 and 30 explore epochs for the two runs, respectively and a seed learning rate of $3e^{-4}$ for both *Knee schedule* and baseline. In all cases we use the model checkpoint with least loss on the validation set for computing BLEU scores on the test set. *Knee schedule* improves the baseline test BLEU score by 0.56 points in the full budget run. In the shorter budget run, *Knee schedule* matches the test accuracy of the baseline schedule while taking 30% less training time. See Figure 9 in Appendix B for training curves.

Table 11: Results for WMT'14 (EN-DE) on Transformer networks. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report mean (stdev) over 3 runs.

LR Schedule	Test BLEU Score	Train Perplexity	Validation Perplexity	Training Epochs
Baseline	27.29 (0.06)	3.87 (0.017)	4.89 (0.02)	70
Knee	27.53 (0.12)	3.89 (0.017)	4.87 (0.006)	70
Knee (short budget)	27.28 (0.17)	4.31 (0.02)	4.92 (0.007)	30

5.3.5 SQuAD-v1.1 fine-tuning on BERT$_{\text{BASE}}$

We also evaluate *Knee schedule* on the task of fine-tuning BERT$_{\text{BASE}}$ model Devlin et al. (2018) on SQuAD v1.1 Rajpurkar et al. (2016) with the Adam Kingma and Ba (2014) opti-

7. `https://github.com/pytorch/fairseq`

Table 12: Training, validation perplexity and test BLEU scores for IWSLT on Transformer networks. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report the mean and standard deviation over 3 runs.

LR Schedule	Test BLEU Score	Train Perplexity	Validation Perplexity	Training Epochs
Baseline	34.97 (0.035)	3.36 (0.001)	4.91 (0.035)	50
Knee	35.53 (0.06)	3.00 (0.044)	4.86 (0.02)	50
Knee (short budget)	35.08 (0.12)	3.58 (0.049)	4.90 (0.063)	35

mizer [8]. BERT fine-tuning is prone to overfitting because of the huge model size compared to the small fine-tuning dataset, and is typically run for only a few epochs. For baseline we use the linear decay schedule mentioned in Devlin et al. (2018). We use a seed learning rate of $3e^{-5}$ and train for 2 epochs. For *Knee schedule*, we train the network with 1 explore epoch with the same seed learning rate of $3e^{-5}$. Table 13 shows our results over 3 runs. We achieve a mean EM score of 81.4, compared to baseline's 80.9, a 0.5% absolute improvement. We don't do a short budget run for this example, as the full budget is just 2 epochs. Please refer to Figure 12 in Appendix B for the training loss, test accuracy and learning rate curves.

Table 13: SQuAD fine-tuning on $\text{BERT}_{\text{BASE}}$. We report the average training loss, and average test EM, F1 scores over 3 runs.

LR Schedule	EM	F1	Train Loss	Training Epochs
Baseline	80.89 (0.15)	88.38 (0.032)	1.0003 (0.004)	2
Knee schedule	81.38 (0.02)	88.66 (0.045)	1.003 (0.002)	2

5.3.6 Result on high accuracy model

To further demonstrate the effectiveness of *Knee schedule*, we took a recent high performing model, Cutoff (Shen et al., 2020)[9], which had reported state-of-the-art accuracy on the IWSLT'14 (DE-EN) dataset. They reported a BLEU score of 37.6 when trained with an inverse square root learning rate schedule for 100 epochs, with the first 6000 steps allocated for warmup. We simply retrained the model with our *Knee schedule*, and achieved a BLEU score of 37.78 (an absolute increase of 0.18). See Table 14 for the BLEU scores, training and validation perplexities.

We also show that *Knee schedule* can train the model in 30% less training time (70 epochs), while achieving slightly better accuracy of 37.66 BLUE score compared to the 100 epoch baseline. The baseline schedule when run for 70 epochs achieves a much worse accuracy of 37.31.

For both the full budget (100 epochs) and the short budget (70 epochs) *Knee* runs, we choose 50% of the total training epochs as explore epochs. We also perform warmup

8. We used the implementation at: `https://github.com/huggingface/transformers`
9. We used the code available at `https://github.com/dinghanshen/Cutoff`

for the same number of steps as baseline. For all runs (*Knee* and baseline), we report the BLEU score obtained by averaging the last 5 checkpoints and computing on the test set. See Figure 10 and 11 in Appendix B for training curves.

Table 14: Training, validation perplexity and test BLEU scores for IWSLT'14 DE-EN on Cutoff. The test BLEU scores are computed by averaging the last 5 checkpoints

LR Schedule	Test BLEU Score	Train Perplexity	Validation Perplexity	Training Epochs
Inv. Sqrt	37.60	3.46	4.24	100
Knee	37.78	3.29	4.13	100
Inv. Sqrt (short budget)	37.31	3.76	4.29	70
Knee (short budget)	37.66	3.48	4.18	70

5.4 Hypothesis Validation with *Knee schedule* on Language Tasks

For validating our hypothesis on the density of wide minima vs narrow minima, we did multiple experiments on vision tasks, most of which were discussed in Section 3. To summarize, in Figures 2 and 3, we showed that for Cifar-10 on Resnet-18, as the number of explore steps increase, the distribution of minima width and test accuracy sharpens and shifts towards wider minima and better accuracy, respectively.

Table 15: IWSLT'14 (DE-EN) on the Transformer network trained with the *Knee schedule*. The *explore* duration is varied, while keeping the total training budget fixed at 50 epochs. We report averages over 3 runs.

Explore Epochs	Test BLEU score	Training Perplexity
5	34.93	3.29
10	35.02	3.22
15	35.08	3.11
20	35.10	3.08
25	35.23	3.02
30	35.28	2.99
40	35.53	3.00

We now perform similar experiments on the IWSLT'14 German to English dataset (Cettolo et al., 2014) trained on Transformer networks (Vaswani et al., 2017) to demonstrate that our hypothesis holds even on a completely different NLP dataset and network architecture. We train with the *Knee schedule* for a total budget of 50 epochs with explore lr as $3e^{-4}$, but keep varying the number of explore epochs. As shown in Table 15, the test BLEU score increases as we increase the number of explore epochs. Further, we found that among multiple trials, a 20 epoch explore run had a high BLEU score of **35.29**, suggesting that the run got lucky. Thus, these results on the IWSLT'14 (DE-EN) dataset add more evidence to the wide-minima density hypothesis.

5.5 Learning Rate Sensitivity for *Knee schedule*

We performed sensitivity analysis of the starting learning rate, referred to as the seed learning rate, for *Knee schedule*. We trained the Cifar-10 dataset on Resnet-18 with the *Knee schedule* for a shortened budget of 150 epochs, starting at different seed learning rates. For each experiment, we do a simple linear search to find the best explore duration. The test accuracies and optimal explore duration for the different seed learning rate choices is shown in Table 16. As shown, the seed learning rate can impact the final accuracy, but *Knee schedule* is not highly sensitive to it. In fact, we can achieve the target accuracy of 95.1 with multiple seed learning rates of 0.05, 0.075, 0.0875 and 0.115, as compared to the original seed learning rate of 0.1, by tuning the number of explore epochs.

Another interesting observation is that the optimal explore duration varies inversely with the seed learning rate. Since a bigger learning rate has higher probability of escaping narrow minima compared to a lower learning rate, it would, on an average, require fewer steps to land in a wide minima. Thus, larger learning rates can *explore* faster, and spend more time in the *exploit* phase to go deeper in the wide minimum. This observation is thus consistent with our hypothesis and further corroborates it.

We also note that by tuning both seed learning rate and explore duration, we can achieve the twin objectives of achieving a higher accuracy, as well as a shorter training time – e.g. here we are able to achieve an accuracy of 95.34 in 150 epochs (seed learning rate 0.075), compared to 95.1 achieved by the baseline schedule in 200 epochs.

Table 16: Seed learning rate sensitivity analysis. Cifar-10 on Resnet-18 trained for 150 epochs with *Knee schedule*. We vary the seed learning rate and explore epochs to get the best test accuracy for the particular setting. We report averages over 3 runs.

Seed LR	Test Accuracy	Optimal Explore Epochs
0.03	95.07	120
0.05	95.12	120
0.0625	95.15	120
0.075	95.34	100
0.0875	95.22	100
0.1	95.14	100
0.115	95.20	60
0.125	95.06	60
0.15	95.04	30

6. Conclusions and Future work

In this paper, we make an observation that an initial *explore* phase with a high learning rate is essential for good generalization of DNNs. Further, we find that a minimum *explore* duration is required even if the training loss stops improving much earlier. We explain this observation via our hypothesis that in the DNN loss landscape, the density of wide minima is significantly lower than that of narrow minima. Motivated by this hypothesis, we present an *Explore-Exploit* based learning rate schedule, called the *Knee schedule*. We do extensive evaluation of *Knee schedule* on multiple models and datasets. In all experiments, the *Knee*

schedule outperforms prior hand-tuned baselines, when trained with the original training budget, and achieves the same test accuracy as the baseline when trained with a much shorter budget.

The explorations in this paper are mostly empirical and an exciting area of future work will be a theoretical analysis of the proposed hypothesis. Another interesting question is related to the densities of wide and narrow minima during stages of training. Currently, our hypothesis on the densities is stated to be true across the entire parameter space. However, most of our observations can be explained even if the hypothesis is true only along the training path of the optimizer. Thus, it will be interesting to analyze the minima densities along the training path vs the landscape in general. The evaluations in Section 5 of *Knee schedule* where we found that the number of *explore* steps consistently help all evaluated optimization schemes, do indicate that the current unconstrained version of the hypothesis is more likely, but a more thorough analysis can be insightful. Another interesting area of future work will be to develop techniques that automatically ascertain that the optimizer has landed in a wide basin, and switch to the exploit portion of *Knee schedule*.

7. Acknowledgement

We would like to thank Sanjith Athlur for his help in setting up the VM cluster for large training runs and Harshay Shah for helpful discussions on minima width computation. We would also like to thank the editor and reviewers for their insightful comments, which helped greatly improve this work.

References

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1):161–170, 2020.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W14/W14-3302`.

Richardh Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 02 2003. doi: 10.1137/0916069.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, page 57, 2014.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint arXiv:1802.10026*, 2018.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *arXiv preprint arXiv:1905.13277*, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Simon Guiroy, Vikas Verma, and Christopher Pal. Towards understanding generalization in gradient-based meta-learning. *arXiv preprint arXiv:1907.07287*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Stanisław Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Stanisław Jastrzebski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SkgEaj05t7`.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical fisher approximation. *arXiv preprint arXiv:1905.12558*, 2019.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *arXiv preprint arXiv:1907.04595*, 2019.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *Advances in Neural Information Processing Systems*, 33:14544–14555, 2020.

Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after sgd reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Levent Sagun, Utku Evci, V Ugur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. iclr 2018 workshop contribution. *arXiv preprint arXiv:1706.04454*, 2017.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*, 2020.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

Piotr A Sokol and Il Memming Park. Information geometry of orthogonal initializations and training. *arXiv preprint arXiv:1810.03785*, 2018.

Brian Swenson, Soummya Kar, H Vincent Poor, José MF Moura, and Aaron Jaech. Distributed gradient methods for nonconvex optimization: Local and global convergence guarantees. *arXiv preprint arXiv:2003.10309*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

Lei Wu, Chao Ma, and E Weinan. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, pages 8279–8288, 2018.

Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *arXiv e-prints*, pages arXiv–2002, 2020.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

Xubo Yue, Maher Nouiehed, and Raed Al Kontar. Salr: Sharpness-aware learning rate scheduler for improved generalization. *arXiv preprint arXiv:2011.05348*, 2020.

Appendix A. Comparisons with Other Baseline Learning Rate Schedules

In this section we compare *Knee schedule* against several other learning rate schedules – one-cycle, linear decay and cosine decay.

One-Cycle: The one-cycle learning rate schedule was proposed in Smith (2018) (also see Smith (2017)). This schedule first chooses a maximum learning rate based on an learning rate range test. The learning rate range test starts from a small learning rate and keeps increasing the learning rate until the loss starts exploding (see figure 4). Smith (2018) suggests that the maximum learning rate should be chosen to be bit before the minima, in a region where the loss is still decreasing. There is some subjectivity in making this choice, although some blogs and libraries[10] suggest using a learning rate one order lower than the one at minima. We go with this choice for all our runs.

Once the maximum learning rate is chosen, the one-cycle schedule proceeds as follows. The learning rate starts at a specified fraction[11] of the maximum learning rate and is increased linearly to the maximum learning rate for 45 percent of the training budget and then decreased linearly for the remaining 45. For the final 10 percent, the learning rate is reduced by a large factor (we chose a factor of 10). We used an opensource implementation [12] for our experiments.

Linear Decay: The linear decay learning rate schedule simply decays the learning rate linearly to zero starting from a seed learning rate.

Cosine Decay: The cosine decay learning rate schedule decays the learning rate to zero following a cosine curve, starting from a seed learning rate.

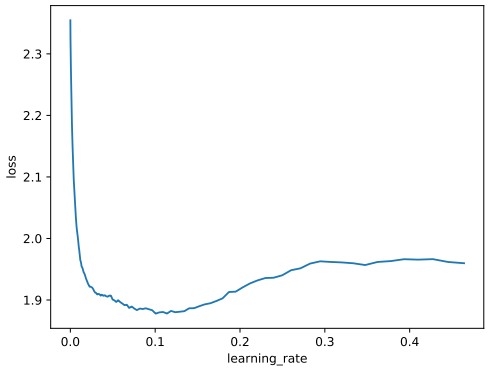
A.1 Cifar-10

Figure 4a shows the learning rate range test for Cifar-10 with the Resnet-18 network. The minima occurs around learning rate of 0.09, and we choose $9e^{-3}$ as the maximum learning rate for the One-Cycle runs. For linear, cosine decay schedules we start with a seed learning rate of 0.1 as used in the standard baselines. The training loss and test accuracy for the various schedules are shown in Table 17 for the full budget runs (200 epochs), and in Table 18 for the short budget runs (150 epochs).

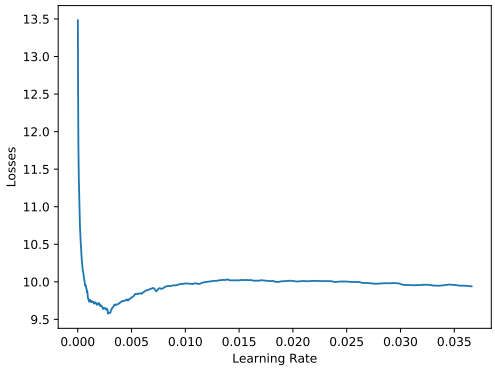
A.2 ImageNet

Figure 4d shows the learning rate range test for ImageNet with the Resnet-50 network. The minima occurs around learning rate of 2.16, and we choose 0.216 as the maximum learning rate for One-Cycle runs. For linear, cosine decay schedules we start with a seed learning rate of 0.1 as used in the standard baselines. The training loss and test accuracy for the various schedules are shown in Table 19 for the full budget runs (90 epochs), and in Table 20 for the short budget runs (50 epochs).

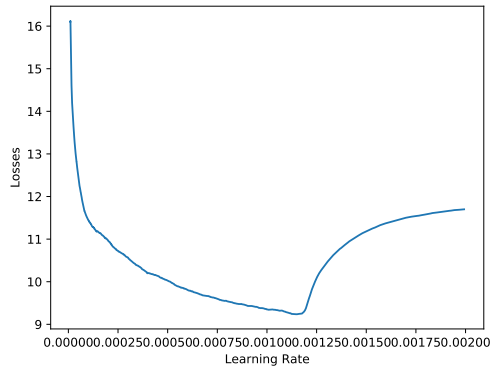
10. See `https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html` and `https://towardsdatascience.com/finding-good-learning-rate-and-the-one-cycle-policy-7159fe1db5d6`. Also see `https://docs.fast.ai/callback.schedule.html#lrfinder` and `https://docs.fast.ai/callback.schedule.html#learner.fit_one_cycle`
11. See div_factor in `https://docs.fast.ai/callback.schedule.html#learner.fit_one_cycle`. We chose the fraction to be 0.1 in our experiments.
12. `https://github.com/nachiket273/One_Cycle_Policy`



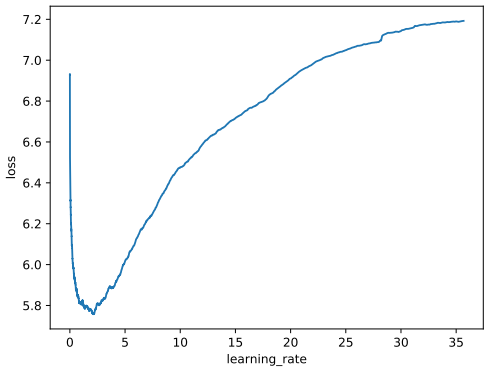
(a) LR range test for CIFAR-10



(b) LR range test for IWSLT'14 DE-EN



(c) LR range test for WMT'14 EN-DE



(d) LR range test for ImageNet

Figure 4: learning rate range test for selecting the maximum learning rate. A good choice is the learning rate is a bit before the minima in a region where the loss is still decreasing.

A.3 WMT'14 EN-DE

Figure 4c shows the learning rate range test for WMT'14 EN-DE on the transformer networks. The minima occurs near $1.25e^{-3}$. For the maximum learning rate, we choose $2.5e^{-4}$ for the default one-cycle policy. For linear, cosine decay schedules we start with a seed learning rate of $3e^{-4}$ as used in the standard baselines The training, validation perplexity and BLEU scores for the various schedules are shown in Table 21 for the full budget runs (70 epochs), and in Table 22 for the short budget runs (30 epochs).

A.4 IWSLT'14 DE-EN

Figure 4b shows the learning rate range test for IWSLT on the transformer networks. The minima occurs near $2.5e^{-3}$. For the maximum learning rate, we choose $2.5e^{-4}$ for the default one-cycle policy. For linear, cosine decay schedules we start with a seed learning rate of $3e^{-4}$ as used in the standard baselines The training, validation perplexity and BLEU scores

for the various schedules are shown in Table 23 for the full budget runs (50 epochs), and in Table 24 for the short budget runs (35 epochs).

A.5 SQuAD-v1.1 finetuning with $\mathbf{BERT_{BASE}}$

We choose $1e^{-5}$ as the maximum learning rate for One-Cycle runs as the minima occurs close to $1e^{-4}$. For linear, cosine decays we start with a seed learning rate of $3e^{-5}$ as used in standard baselines. Table 25 show the average training loss, average test EM and F1 scores for the various schedules. We did not do a short budget training for this dataset, as the full budget is just 2 epochs.

Table 17: Cifar-10 on Resnet-18 full budget training (200 epochs): Training loss and Test accuracy for more learning rate schedules. We report the mean and standard deviation over 7 runs.

LR Schedule	Test Accuracy	Train Loss
One-Cycle	94.08 (0.07)	0.0041 (6e-5)
Cosine Decay	95.23 (0.11)	0.0023 (9e-5)
Linear Decay	95.18 (0.15)	0.0018 (7e-5)
Knee schedule	**95.26** (0.11)	0.0023 (1e-4)

Table 18: Cifar-10 on Resnet-18 short budget training (150 epochs): Training loss and Test accuracy for more learning rate schedules. We report the mean and standard deviation over 7 runs.

LR Schedule	Test Accuracy	Train Loss
One-Cycle	93.84 (0.082)	0.0052 (7e-5)
Cosine Decay	95.06 (0.16)	0.0030 (2e-4)
Linear Decay	95.02 (0.10)	0.0021 (1e-4)
Knee schedule	**95.14** (0.18)	0.0044 (3e-4)

Table 19: ImageNet with ResNet-50 full budget training (90 epochs): Training loss, Test Top-1 and Test Top-5 for more learning rate schedules. We report the mean and standard deviation over 3 runs.

LR Schedule	Test Top-1	Test Top-5	Train Loss (av)
One Cycle	75.39 (0.137)	92.56 (0.040)	0.96 (0.003)
Cosine Decay	76.41 (0.212)	93.28 (0.066)	0.80 (0.002)
Linear decay	76.54 (0.155)	93.21 (0.051)	0.75 (0.001)
Knee schedule	**76.71** (0.097)	**93.32** (0.031)	0.79 (0.001)

Table 20: ImageNet with ResNet-50 short budget training (50 epochs): Training loss, Test Top-1 and Test Top-5 for more learning rate schedules. We report the mean and standard deviation over 3 runs.

LR Schedule	Test Top-1	Test Top-5	Train Loss (av)
One Cycle	75.36 (0.096)	92.53 (0.079)	1.033 (0.004)
Cosine Decay	75.71 (0.116)	92.81 (0.033)	0.96 (0.002)
Linear decay	75.82 (0.080)	92.84 (0.036)	0.91 (0.002)
Knee schedule	**75.92** (0.11)	**92.90** (0.085)	0.90 (0.003)

Table 21: WMT'14 (EN-DE) on Transformer networks full budget training (70 epochs): Training, validation perplexity and test BLEU scores for more learning rate schedules. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report the mean and standard deviation over 3 runs.

LR Schedule	Test BLEU Score	Train ppl	Validation ppl
One-Cycle	27.19 (0.081)	3.96 (0.014)	4.95 (0.013)
Cosine Decay	27.35 (0.09)	3.87 (0.011)	4.91 (0.008)
Linear Decay	27.29 (0.06)	3.87 (0.017)	4.89 (0.02)
Knee schedule	**27.53** (0.12)	3.89 (0.017)	4.87 (0.006)

Table 22: WMT'14 (EN-DE) on Transformer networks short budget training (30 epochs): Training, validation perplexity and test BLEU scores for more learning rate schedules. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report the mean and standard deviation over 3 runs.

LR Schedule	Test BLEU Score	Train ppl	Validation ppl
One-Cycle	26.80 (0.2)	4.38 (0.017)	5.02 (0.007)
Cosine Decay	26.95 (0.23)	4.32 (0.013)	4.99 (0.011)
Linear Decay	26.77 (0.12)	4.36 (0.092)	5.02 (0.01)
Knee schedule	**27.28** (0.17)	4.31 (0.02)	4.92 (0.007)

Table 23: IWSLT'14 (DE-EN) on Transformer networks full budget training (50 epochs): Training, validation perplexity and test BLEU scores for more learning rate schedules. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report the mean and standard deviation over 3 runs.

LR Schedule	Test BLEU Score	Train ppl	Validation ppl
One-Cycle	34.77 (0.064)	3.68 (0.009)	4.97 (0.010)
Cosine Decay	35.21 (0.063)	3.08 (0.004)	4.88 (0.014)
Linear Decay	34.97 (0.035)	3.36 (0.001)	4.92 (0.035)
Knee schedule	**35.53** (0.06)	3.00 (0.044)	4.86 (0.02)

Table 24: IWSLT'14 (DE-EN) on Transformer networks short budget training (35 epochs): Training, validation perplexity and test BLEU scores for more learning rate schedules. The test BLEU scores are computed on the checkpoint with the best validation perplexity. We report the mean and standard deviation over 3 runs.

LR Schedule	Test BLEU Score	Train ppl	Validation ppl
One-Cycle	34.43 (0.26)	3.98 (0.028)	5.09 (0.017)
Cosine Decay	34.46 (0.33)	3.86 (0.131)	5.06 (0.106)
Linear Decay	34.16 (0.28)	4.11 (0.092)	5.14 (0.066)
Knee schedule	**35.08** (0.12)	3.58 (0.063)	4.90 (0.049)

Table 25: SQuAD-v1.1 fine-tuning on $\text{BERT}_{\text{BASE}}$ for more learning rate schedules. We report the average training loss, average test EM, F1 scores over 3 runs.

LR Schedule	EM (av)	F1 (av)	Train Loss (av)
One Cycle	79.9 (0.17)	87.8 (0.091)	1.062 (0.003)
Cosine Decay	81.31 (0.07)	88.61 (0.040)	0.999 (0.003)
Linear decay	80.89 (0.15)	88.38 (0.042)	1.0003 (0.004)
Knee schedule	**81.38** (0.02)	**88.66** (0.045)	1.003 (0.002)

Appendix B. Detailed Plots

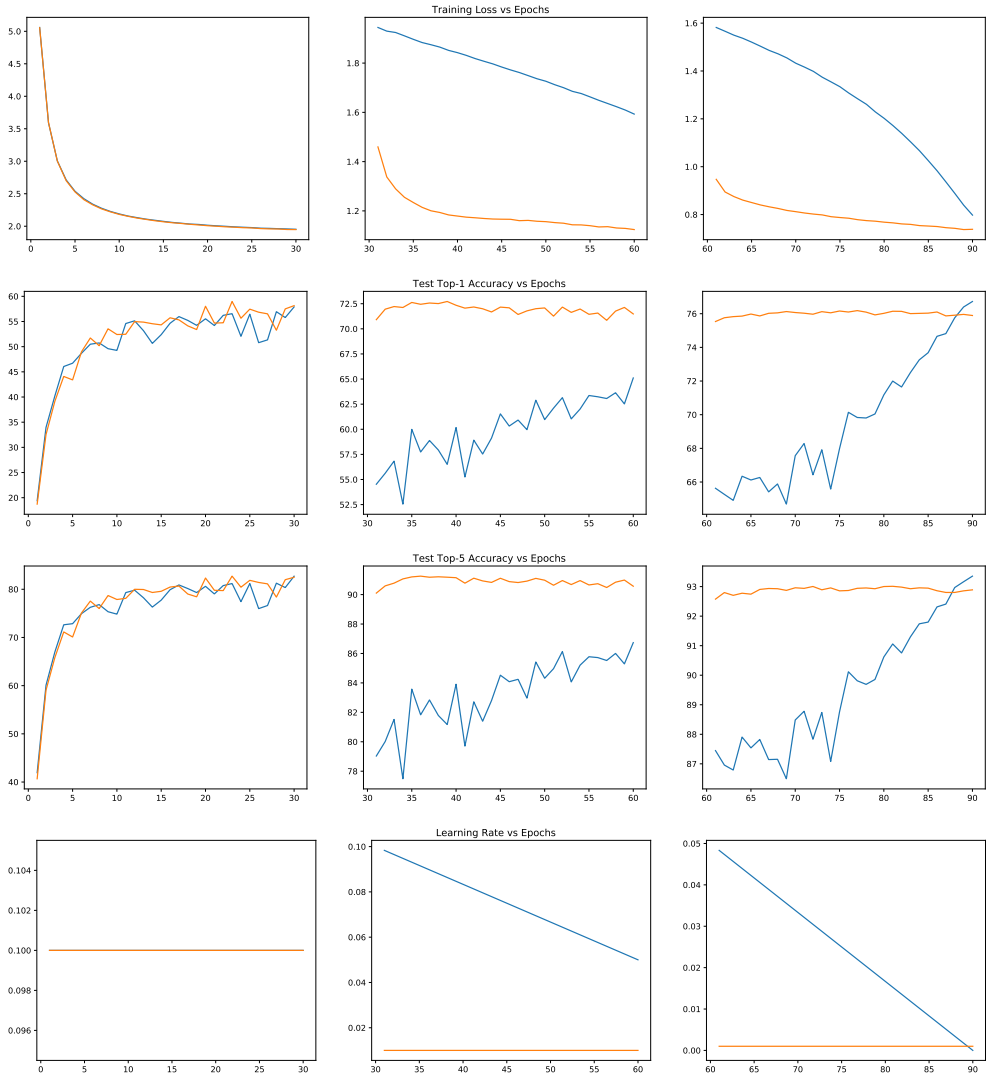


Figure 5: ImageNet on Resnet-50 trained with Momentum. Shown are the training loss, top-1/top-5 test accuracy and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue). The plot is split into 3 parts to permit higher fidelity in the y-axis range.

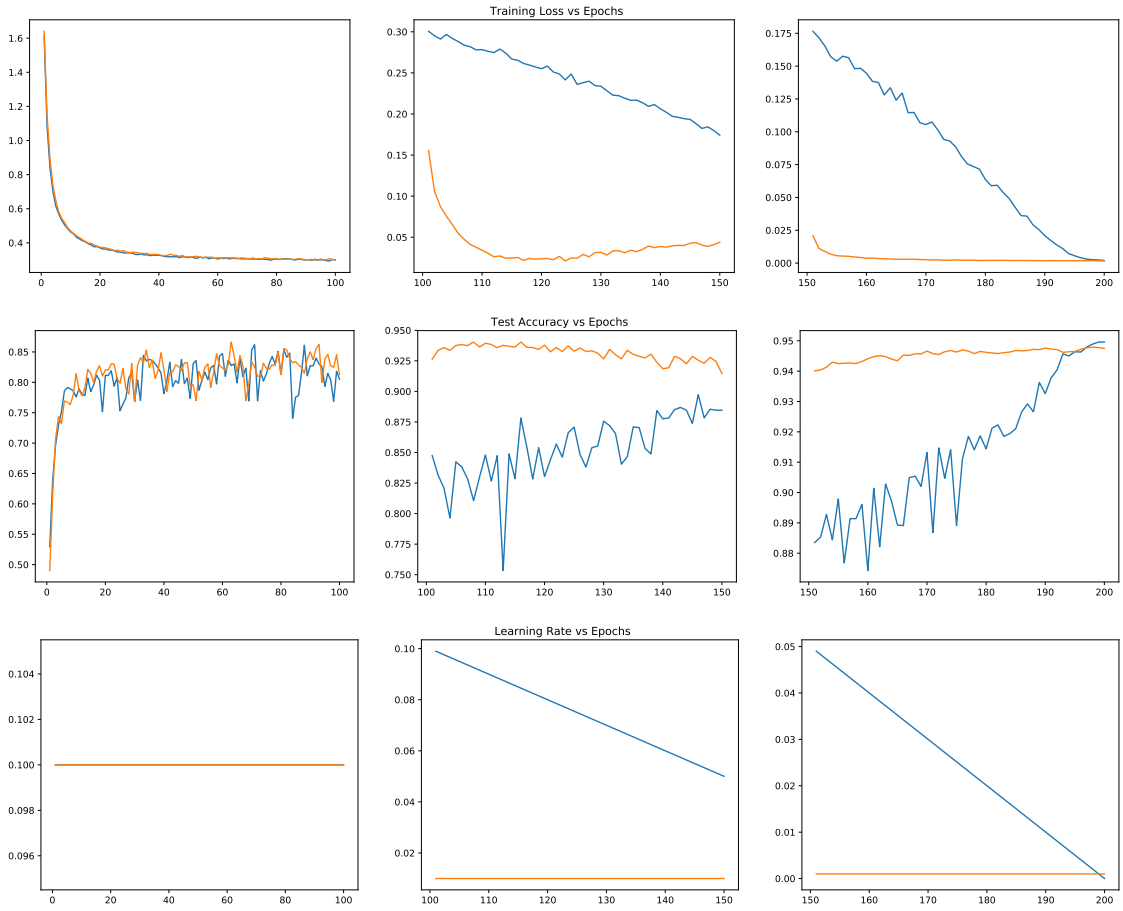


Figure 6: Cifar-10 on Resnet-18 trained with Momentum. Shown are the training loss, test accuracy and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue). The plot is split into 3 parts to permit higher fidelity in the y-axis range.

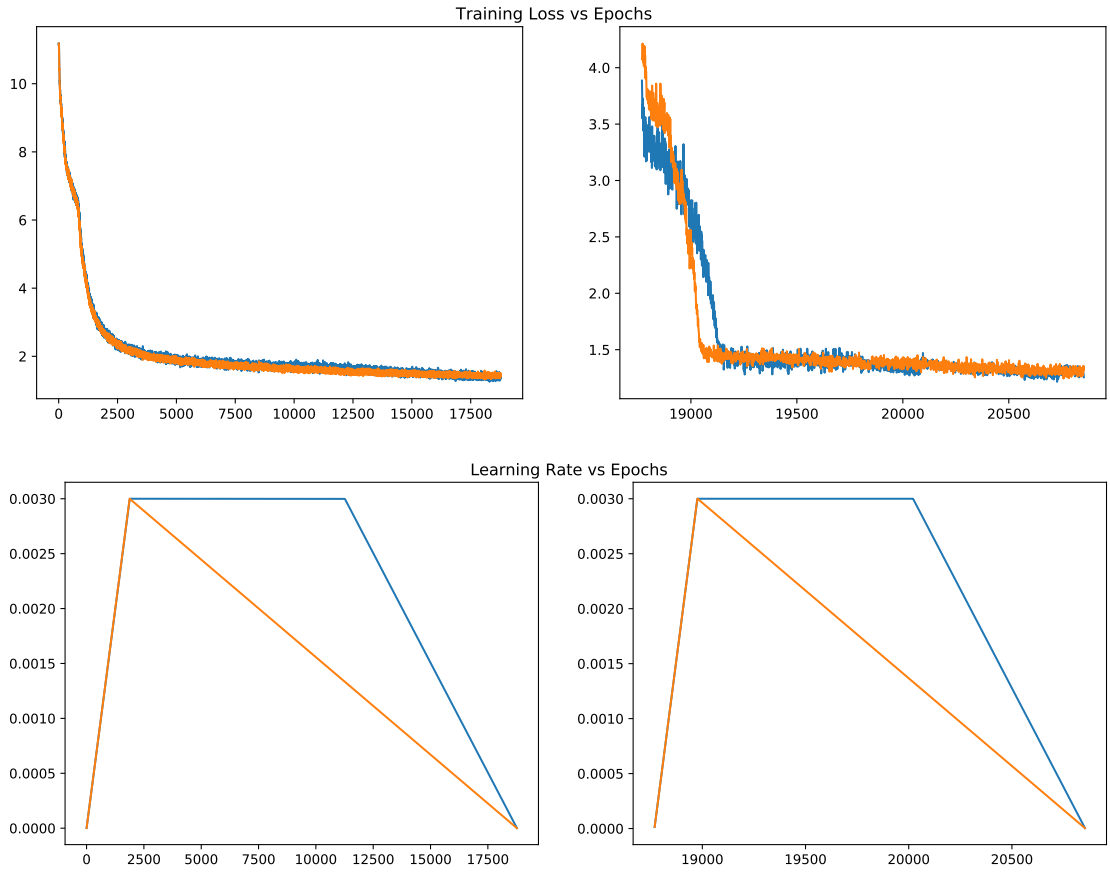


Figure 7: $\text{BERT}_{\text{LARGE}}$ pretraining for batch size of 16k with LAMB optimizer for the short budget runs. Shown are the training loss and learning rate as a function of steps, for the baseline scheme short budget (orange) vs the *Knee schedule* scheme short budget (blue). The plot is split into 2 parts to give a clear picture of the two phases of training Devlin et al. (2018). Note that even though the training loss curves look similar for the two runs, we see a significant gap in F1 score obtained when we fine-tune the model checkpoints on SQuAD-v1.1 Rajpurkar et al. (2016). See Table 26 for details.

LR Schedule	F1 - Trial 1	F1 - Trial 2	F1 - Trial 3	F1 avg.	F1 max
Baseline (short budget)	90.39	90.64	90.53	90.52	90.64
Knee schedule (short budget)	91.22	91.29	91.18	91.23	91.29
Knee schedule (full budget)	91.45	91.41	91.51	91.46	91.51

Table 26: SQuAD fine-tuning on $\text{BERT}_{\text{LARGE}}$. We report F1 scores for 3 different trials as well as the maximum and average values.

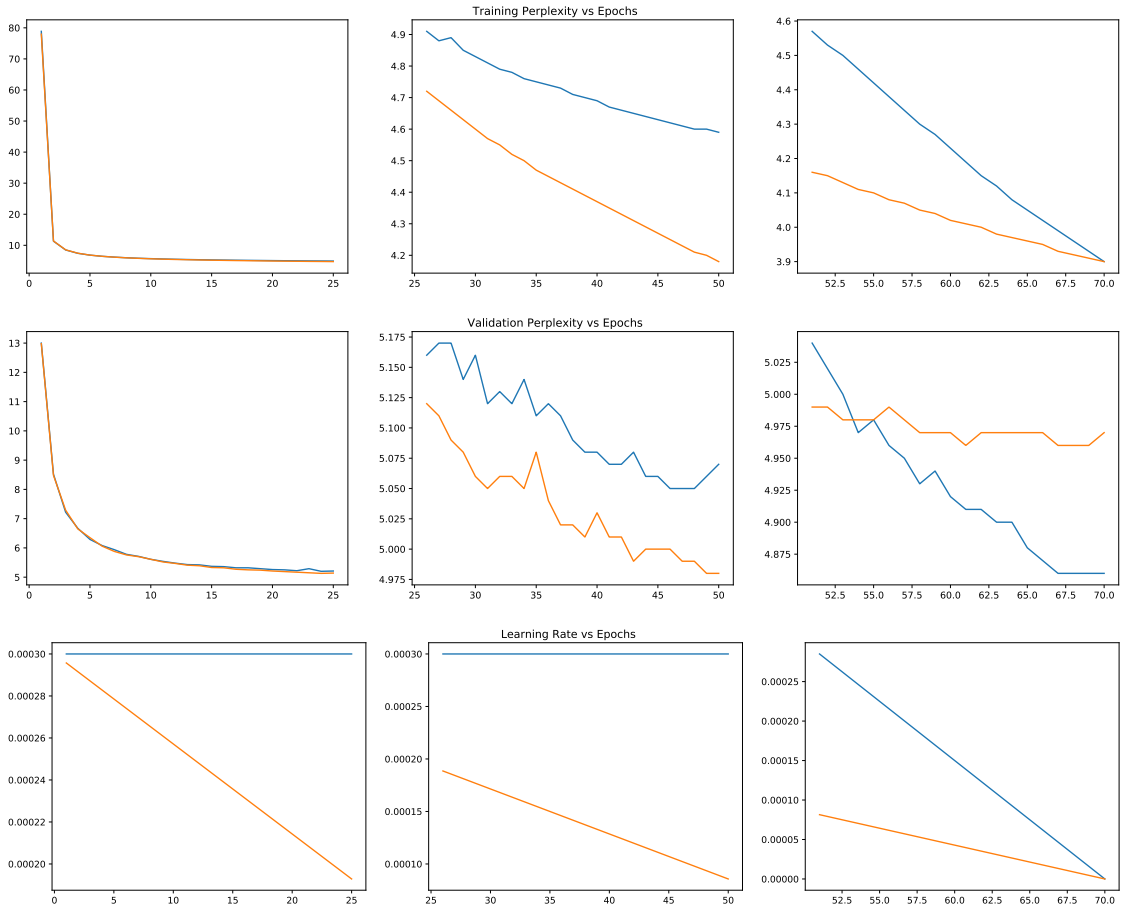


Figure 8: WMT'14 (EN-DE) on Transformer$_{\text{BASE}}$ network trained with RAdam. Shown are the training perplexity, validation perplexity and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue). The plot is split into 3 parts to permit higher fidelity in the y-axis range.

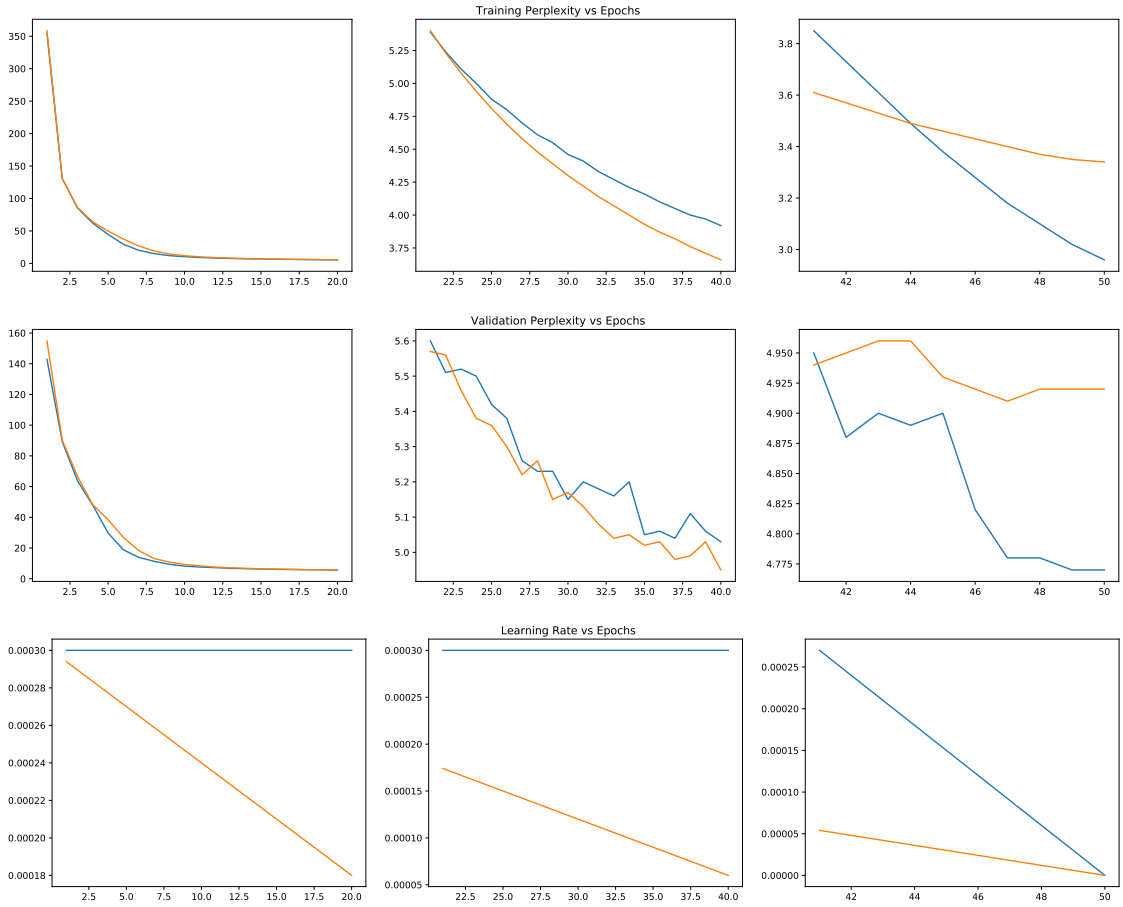


Figure 9: IWSLT'14 (DE-EN) on Transformer$_{\text{BASE}}$ network trained with RAdam. Shown are the training perplexity, validation perplexity and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue). The plot is split into 3 parts to permit higher fidelity in the y-axis range.

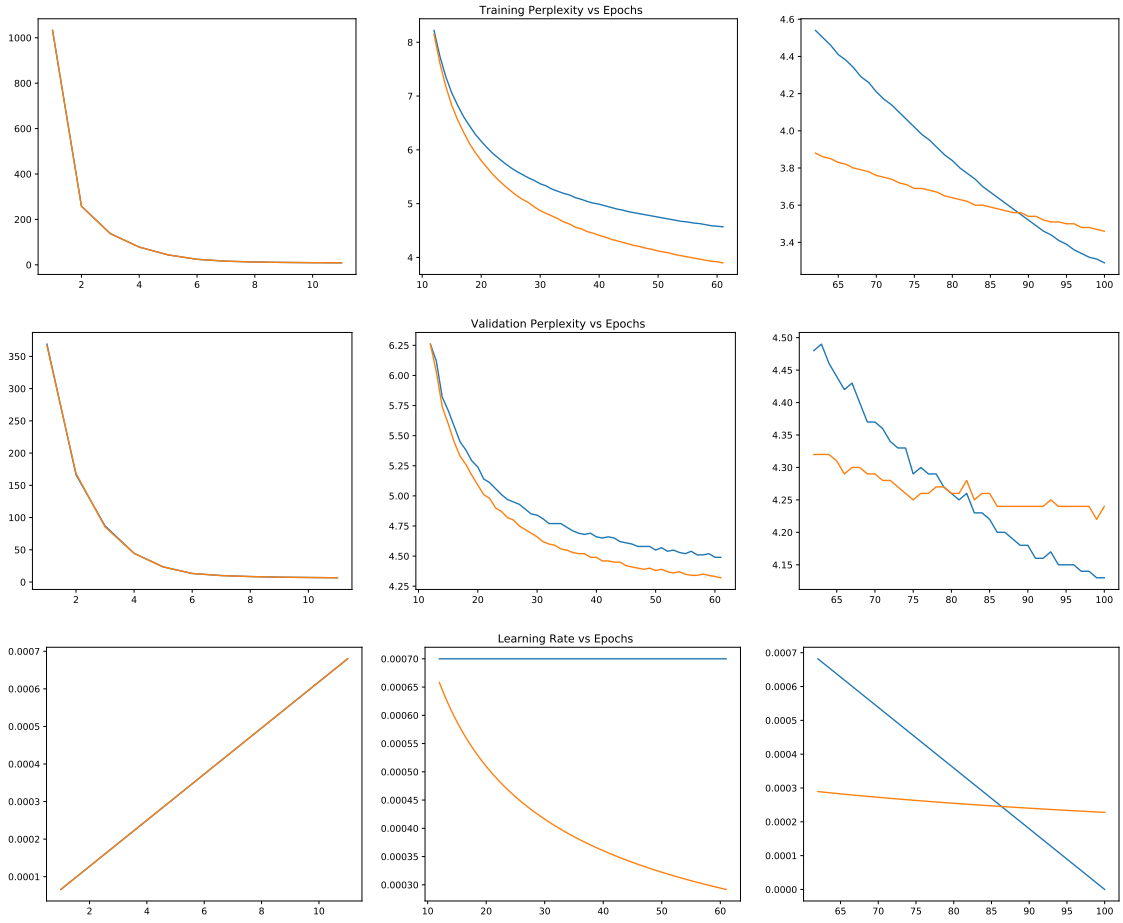


Figure 10: IWSLT'14 (DE-EN) on the model Cutoff (Shen et al., 2020), trained with Adam. Shown are the training perplexity, validation perplexity and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue).

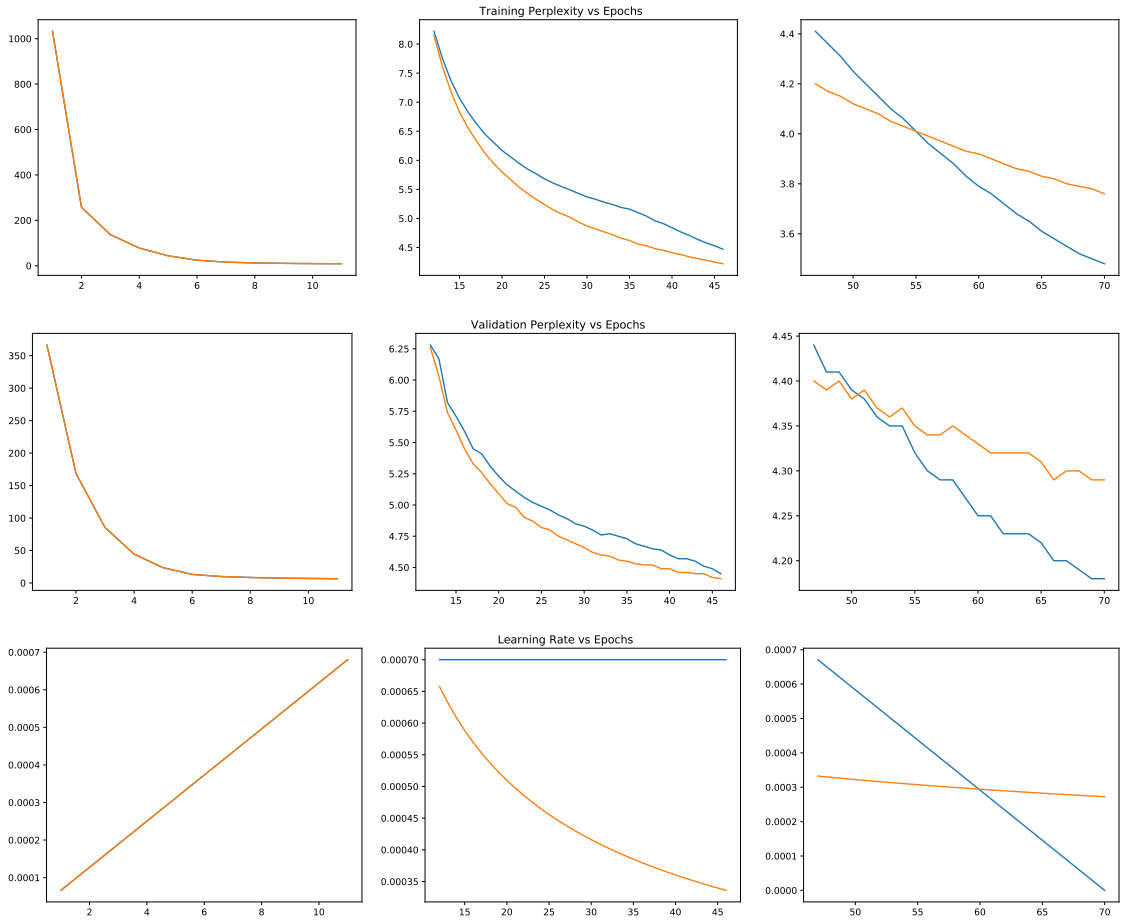


Figure 11: IWSLT'14 (DE-EN) on the model Cutoff (Shen et al., 2020), trained with Adam with a reduced training budget of 70 epochs. Shown are the training perplexity, validation perplexity and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue).

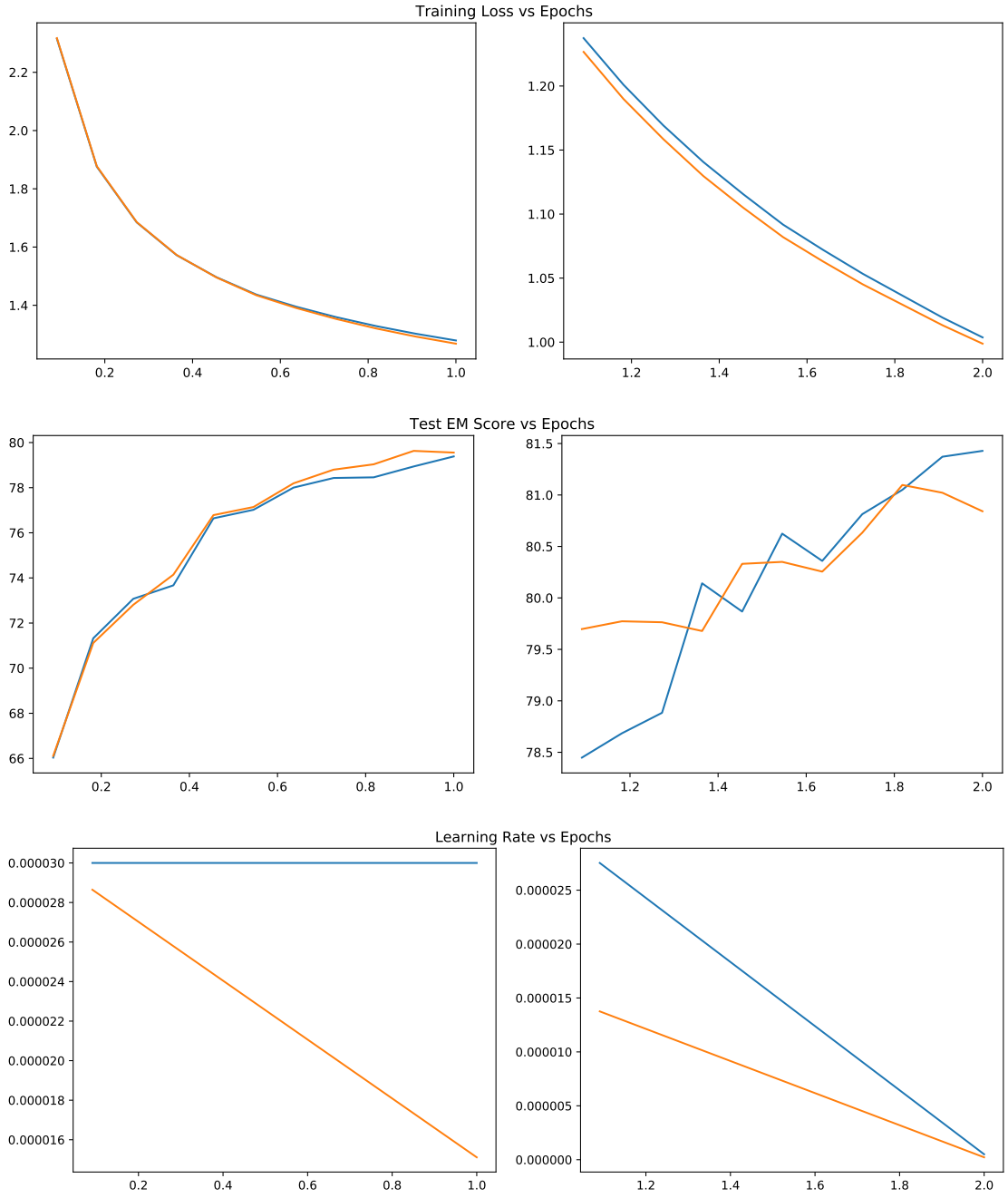


Figure 12: SQuAD-v1.1 fine-tuning on $\text{BERT}_{\text{BASE}}$ trained with Adam. Shown are the training loss, test EM score, and learning rate as a function of epochs, for the baseline scheme (orange) vs the *Knee schedule* scheme (blue). The plot is split into 2 parts to permit higher fidelity in the y-axis range. It is clear that with *Knee schedule* the network starts to overfit after the 2nd epoch, where the testing loss continues to go down, but generalization suffers. We saw similar behavior with different seeds, and thus need to train with *Knee schedule* for only 2 epochs.