Optimization Matters: Guidelines to Improve Representation Learning with Deep Networks

Aline R. Becher[1], **Moacir A. Ponti**[1]

[1]Instituto de Ciências Matemáticas e Computação
Universidade de São Paulo, São Carlos/SP 13566–590, Brazil

becher.aline@usp.br,moacir@icmc.usp.br

Abstract. *Training deep neural networks is a relevant problem with open questions related to convergence and quality of learned representations. Gradient-based optimization methods are used in practice, but cases of failure and success are still to be investigated.In this context, we set out to better understand the convergence properties of different optimization strategies, under different parameter options. Our results show that (i) feature embeddings are impacted by different optimization settings, (ii) suboptimal results are achieved by the use of default parameters, (iii) significant improvement is obtained by making educated choices of parameters, (iv) learning rate decay should always be considered. Such findings offer guidelines for training and deployment of deep networks.*

1. Introduction

Representation learning is the ability to infer a model capable of transforming an input space into a representation of it, which can be, for example, a vector space of characteristics that are relevant to a particular task or application [Bengio et al. 2013]. There is a known potential in Deep Neural Networks (DNN) to find useful representations of signals and images [Ponti et al. 2017]. However, there are challenges related to the convergence and generalization of the parameters of models [Bottou et al. 2018, Zhang et al. 2016].

Optimizing parameters of Convolutional Neural Networks (CNNs), the most relevant architecture for classification, is not trivial due to the non-linear and non-convex characteristics, the high dimensionality of parameters, the presence of local minima and saddle points, as well as lack of theoretical guarantees of convergence [Goodfellow et al. 2016]. Also, even for small networks, training is still known to be NP-hard [Bottou et al. 2018]. Therefore it is paramount to make educated choices of optimization methods and training strategies [Ponti et al. 2021]. Many studies focus on finding the best network architectures, while neglecting the optimization details, with unjustified or arbitrary choices, e.g. using default values. In this context, investigating the behaviour of parameter optimization methods is of critical importance to obtain an improvement of the models for learning general representations.

The most popular class of optimization algorithms are gradient-based methods [Andrade et al. 2018, Serpa et al. 2019, Rodrigues et al. 2017, Abello and Hirata 2019], but there is lack of empirical evidence on cases of failure and success in terms of optimization choices. This motivated the use of computational power to find adequate parameters by meta-heuristics or grid-search [Domhan et al. 2015, Zhang et al. 2015, Miikkulainen et al. 2019], since the

choices are dependent on the network architecture and the initialization or starting point [Li et al. 2018].

The classic Gradient Descent algorithm [Cauchy 1847] calculates the gradient over all samples of the training set, while stochastic versions are faster by using a single training instance (or a batch). The Stochastic Gradient Descent (SGD) [Robbins and Monro 1951] have good convergence results for both convex and non-convex functions, but have high variance stretched algorithms fluctuate between local minima, which may be inefficient. Although there are methods to reduce variance, such as Stochastic Averaged Gradient (SAG) [Schmidt et al. 2017] and SAGA [Defazio et al. 2014], interestingly, those are not often used in the recent literature. Instead, another class of gradient-based algorithms that applies strategies to control the step size during optimization are much more popular, including Adaptive Gradient Algorithm (Adagrad) [Duchi et al. 2011] that adapts the learning rate according to the parameters under optimization by storing the gradient history for parameters: a smaller step is defined for more common parameters, and larger ones for those which characteristics are less frequent. The spread of Root Mean Square (RMSprop) [Tileman and Hinton 2012] was proposed to improve over Adagrad. However, its main assumption is difficult to hold, i.e. the signal of the gradient is similar for examples in a batch [Bottou et al. 2018]. The Adaptive moment estimation (Adam) [Kingma and Ba 2015] combines benefits of both Adagrad and RMSprop, without unrealistic assumptions. But it stills has high variance in the early stages of training. To try to get around this problem, Rectified Adam (RAdam) [Liu et al. 2020] proposes a warm-up term to reduce such early variance. It is noteworthy that nearly every state-of-the-art computer vision model was trained using regular SGD, including AlexNet [Krizhevsky et al. 2012], VGGNet [Simonyan and Zisserman 2014], ResNet [He et al. 2016], SqueezeNet [Iandola et al. 2016], Faster R-CNN [Ren et al. 2015], Single Shot Detectors [Liu et al. 2015] and RetinaNet [Lin et al. 2017]. Thus, each gradient-based algorithm has advantages and disadvantages and the choice of the algorithm and their setup for a given problem is still a matter of investigation, with the complexity of the problem raising questions [Sun 2019]. Concerns about overfitting and regularization as well as the impact of initialization [Sutskever et al. 2013, Le et al. 2011] still lack answers.

In this paper, we analyze in practice the behaviour of SGD, Adam, and its variation RAdam in light of feature learning. As far as we know, this is the first work focusing on how optimization choices impact the discriminative power of such representations including scenarios of transfer learning. By exploring CNNs and AEs, as well as popular algorithms and tasks, we offer relevant discussion on why optimization matters and how to narrow down choices depending on the task at hand.

2. Related Work and Contributions

Some recent studies aimed at analyzing a large number of optimizers and possible parameter choices [Wilson et al. 2018, Choi et al. 2019, Sivaprasad et al. 2019, Schmidt et al. 2020]. In [Wilson et al. 2018] the authors found that Adam, Adagrad and RMSprop converged to worse solutions than SGD and GD, but for general applications. Later, [Choi et al. 2019] proposed a comparison protocol with manual choice of hyperparameters. The SGD, Momentum, Nesterov, Adam, RMSprop and Nadam optimizers

are evaluated. They concluded that RMSprop and Adam never underperformed SGD, Momentum and Nesterov methods. Along similar lines, [Sivaprasad et al. 2019] evaluated natural image datasets and architectures taking into account the computational cost to find such parameters. They concluded Adam works best without much tuning, while SGD may have superior performance but require finding its proper configuration. More recently, [Schmidt et al. 2020] also evaluated multiple optimizers and hyperparameters. They could not identify which optimizer had the better performance, pointing out the importance of narrowing down the hyperparameter search space.

This paper does not intent to find the best optimization algorithm. Instead, we start by introducing a formulation for optimizing deep networks, the algorithms under study, and then show empirical evidence that shed light on **improving the process of feature learning by relating different choices of optimizer, batchsize and learning rate value/schedule**. Therefore, our contributions are mainly:

I. evaluate performance with 3 optimizers considering different values of batch size, learning rates choices and decay strategies, as well as different datasets;
II. showing how standard choices usually result in poor generalization capacity;
III. defining guidelines for the appropriate selection of parameters when training from scratch and for finetuning of pre-trained models to improvement generalization and discriminative capacity of the learned representations.

3. Theoretical background of deep learning optimization

3.1. SGD (Stochastic Gradient Descent)

SGD is an approximation of GD that allow calculating the gradient of the cost function based on a single example or a small subset (minibatch), X_k, selected for the iteration k. The parameters are updated as follows:

$$\theta_{k+1} = \theta_k - \alpha_k g(X_k, \theta_k),$$

where $g(.)$ is the gradient of the cost function. It has theoretical results for convex and non-convex functions, ensuring that the sequence of the iterates generated by the algorithm has gradients that converge to zero. Also, the sum of the squares remains finite, which is a necessary condition for convergence.

3.2. Adam

Adam calculates individual learning rates for different parameters using first and second gradient estimates, as follows:

$$\theta_k = \theta_{k-1} - \alpha_k \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon},$$

where $\hat{m}$ and $\hat{v}$ are corrected estimates of the first and second moment of the gradient, respectively. Adaptive methods also have theoretical results of convergence, ensuring that the gradient norm remains small in a finite number of iterations and converges to a critical point.

3.3. RAdam

RAdam is a variation of Adam that adjusts the adaptive learning rate to correct the effects of the large variance in the initial training stage, since it can result in bad local minima at the beginning of the training with Adam or SGD. A warm-up term seeks a learning rate to avoid falling into bad local minima. Since this varies according to the data set, its adaptive approach shown to improve results over Adam.

4. Experiments

Our experiments evaluate feature spaces learned by CNNs when using as optimizes SGD, Adam and RAdam, under different learning rates (LR) and batch sizes. Note we are not interested in beating state-of-the-art results, but rather studying the effects of such choices in learning. We focus on representation learning, studying feature spaces obtained from the trained models. The Support Vector Machine (SVM) classifier is used to evaluate the linear discriminative capability of such spaces since it is the classifier with the strongest bias and learning guarantees [Mello and Ponti 2018]. Therefore, the SVM (no kernels, and cost=1), is trained on the feature spaces learned by the CNNs (output of the layer before prediction). We use the original training/test split of the datasets.Experiments were repeated 5 times with the same 5 fixed seeds for all models (11, 27, 44, 86 and 104), which also test how each optimization setup behaves under different starting points.

The initial experiments use a smaller CNN architecture for Cifar-10 and Fashion-MNIST with fully connected and convolutional layers. Each experiment is a variation of:

- Datasets: Fashion-MNIST and Cifar-10;
- Batch sizes: 8, 16, 32, 64, 128, 256 and 512
- Learning Rates (LR): 0.5, 0.1, 0.01 and 0.001.

Later we also explored ResNet-50 and MobileNetV2, both with random weights and pre-trained in ImageNet to investigate optimization on transfer learning and Learning Rate scheduling approaches: Step Decay (SD), which consists of decreasing the learning rate by a factor of 0.5 every five epochs and Exponential Decay (ED), which consists of drop learning rate by an exp factor $exp(-0.1)$. In summary, those experiments evaluated:

- Architectures: Resnet50 and MobileNetV2;
- Optimizers: SGD and RAdam;
- Datasets: Cifar-10 and Flowers;
- Batch sizes: 32, 64, 128, 256;
- Learning Rates: 0.5, 0.1, 0.01 (SGD) and 0.1, 0.01, 0.001 and 0.0001 (RAdam);
- Weights: Random Initialization and Weights Pre-Trained from Imagenet;
- Scheduling: Step Decay (SD) and Exponential Decay (ED).

5. Results and discussion

5.1. Evaluating separability of feature spaces

– **Fashion-MNIST classification** (Table 1): all methods achieved $\sim 92\%$ accuracy, but different configurations are required. SGD require higher learning rates, e.g. 0.5 and 0.1, and batch sizes from 16 to 64. Except for $LR = 0.5$, the accuracy decreases as the batch size increases. For Adam and RAdam, smaller learning rates, e.g. 0.001 and

larger batch sizes, up to of 512, are allowed and may improve results. Overall, SGD was more robust regarding learning rate variation, while Adam/RAdam are negatively affected higher learning rates.

– **Cifar-10 classification** (Table 2): SGD is also favoured by larger learning rates and smaller batch sizes (even 8). Adam/RAdam were overral better than SGD, with smaller learning rates and large batch sizes preferable.

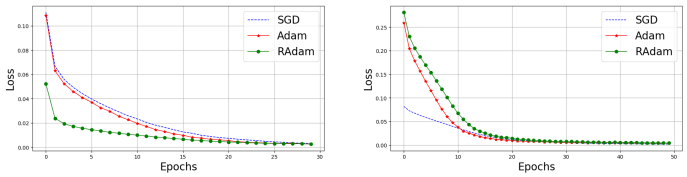
Tabela 1. Classification accuracy (%) for Fashion Mnist features comparing learning rates and batch sizes. Values in bold are accuracies of 91 or higher.

LR	OPT	Batchsize						
		8	**16**	**32**	**64**	**128**	**256**	**512**
	SGD	18.6±17	**91±0.2**	**91.5±0.2**	**91.7±0.2**	**91.4±0.3**	**91 ±0.2**	90.2±0.5
0.5	**Adam**	10±0	10±0	10±0	10±0	10±0	10 ±0	10±0
	RAdam	10±0	10±0	10±0	10±0	10±0	10 ±0	10±0
	SGD	**91.7±0.2**	**91.6±0.2**	**91.3±0.3**	**91.1± 0.3**	90.1±0.6	88.9±0.6	87.7±0.3
0.1	**Adam**	10±0	10±0	10±0	10±0	10±0	10±0	10±0
	RAdam	10±0	10.3±0.6	10±0	10±0	10±0	10±0	10.5±1
	SGD	90.8±0.5	89.7±0.5	88.9±0.3	88.1 ±0.3	87±0.2	85.5±0.3	83.5±0.6
0.01	**Adam**	10±0	10±0	16.1±6.5	17.8±15.6	24.3 ±22.3	58.3±3.4	73.6±5.7
	RAdam	10±0	10±0	10.9±1.9	11.8±3.7	10±0	10±0	58.6±4.2
	SGD	87.7±0.3	86.5±0.3	85±0.4	82.5 ±0.7	78.2± 2.2	60.2±14.9	35.8±22.8
0.001	**Adam**	56.6±6.9	53.9± 2.5	79± 2	90.3±0.9	**91.4±0.1**	**91.8±0.3**	**91.9±0.1**
	RAdam	52.8±9.2	52.9±6.8	75.7±3.5	90.6±0.3	**91.7±0.1**	**91.5±0.3**	**91.7±0.2**

Tabela 2. Classification accuracy (%) for Cifar-10 features comparing learning rates and batch sizes. Values in bold are accuracies of 61 or higher.

LR	OPT	Batchsize						
		8	**16**	**32**	**64**	**128**	**256**	**512**
	SGD	10.5±0.6	10.2±0.3	59.4 ±0.8	**61.5±1.1**	**61.4±2**	60.5 ±1.8	57.5±3.5
0.5	**Adam**	10±0	10±0	10±0	10± 0	10± 0	10±0	10±0
	RAdam	10±0	10±0	10±0	10±0	10± 0	10±0	10±0
	SGD	**63±0.7**	**63±0.5**	**62.3±0.8**	**61.2 ±1.4**	60.3±1.7	59.1±1.9	56.4±1.4
0.1	**Adam**	10±0	10±0	10±0	10± 0	10± 0	10±0	10±0
	RAdam	10±0	10±0	10±0	10±0	10± 0	10±0	10±0
	SGD	**61.3±1.5**	60.5 ±1.4	59.8±1.4	56±0.9	50.8±0.8	45.7±0.9	39.5±1.5
0.01	**Adam**	10±0	10±0	10±0	10±0	18.1±16.2	16.8±30.7	17±14
	RAdam	10±0	10±0	10±0	10±0	10±0	10±0	55.5±1
	SGD	53.9±0.9	49.2±0.9	43.7±0.9	37.1±1.9	30.1±3.2	21.3±6	15.7±6.6
0.001	**Adam**	54.1±2.2	54.2±2.6	52.1±2.3	60.3±1.8	**63.9±1.1**	**64.4±0.6**	**64.3±0.4**
	RAdam	56.9±0.9	54.4±0.9	51.3±2.7	59.2±1.2	**63.3±0.6**	**64.3±0.6**	**63.9±1**

The loss values over training epochs for Fashion-MNIST and Cifar-10 are shown in Figure 1. Using the best choice of the learning rate and batch size: RAdam allows for faster convergence when the size of the network is adequate to the problem (Fashion), but slower for harder problems (Cifar-10).



Fashion-MNIST Cifar-10

Figura 1. Training loss for Fashion-MNIST (left) and Cifar-10 (right) datasets comparing optimizer's best results.

t-SNE Visualization (Figure 2): evaluates projection of Fashion-MNIST test data by varying the LR and optimizer choices and fixing the best batch size (64 for SGD; 256 for Adam/RAdam). This confirms SGD is less sensitive with respect to learning, while Adam/RAdam require lower learning rates, failing to produce a reasonable feature space for wrong choices, which includes LR=0.01, the default for many deep learning frameworks.

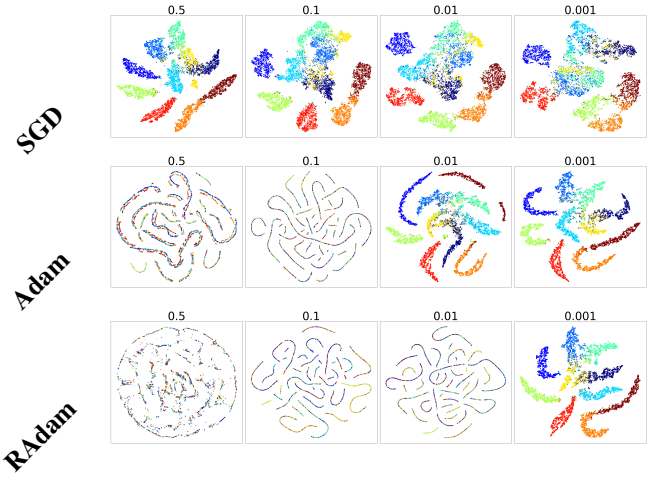


Figura 2. Projection of the characteristics space with t-SNE comparing the different learning rates, with CNN architecture and Fashion Mnist dataset considering the best results over all batch sizes.

5.2. A case-study with state-of-the-art CNNs

In this section we used a Residual Network with 50 layers (ResNet50) [He et al. 2016] an a MobileNetV2 [Sandler et al. 2018], often used to perform Transfer Learning.

First, using ResNet50, the top layer was removed and replaced by four dense layers (with size 1024, 512, 256 and 128) with Dropout (0.2) every two layers, followed by a softmax classification layer with 10 classes. After training from scratch for 50 epochs with Cifar-10, as expected, the results improve achieving up to 84% accuracy (see Table 3). From pre-trained weights, with 25 epochs a $\sim 82\%$ accuracy is achieved. Extreme choices impact less SGD, e.g. see SGD with LR=0.01 and RAdam with LR=0.1, but both are able to achieve similar accuracies with appropriate choices. For pre-trained weights, it is necessary to start with a smaller step. There seems to exist a sweet spot: 0.1 for SGD and 0.001 for RAdam, which resulted in slightly better accuracies than SGD.

The MobileNet was used for the Cifar-10 and Flowers Recognition datasets. The top of the net was removed and replaced by a dense layer (with size 128) followed by a softmax classification layer. As observed with the Resnet50 architecture, the learning rates 0.1 for SGD and 0.001 for RAdam achieved the best results. With the Flowers Recognition dataset, SGD behaves differently than we have seen before, but this time favoured by lower learning rates, as shown in Table 5. This is because Flowers is a fine-grained dataset so, in this case the general behavior is not repeated.

In all cases, the more LR is far from its ideal value, the more a large batch size degrades results. Thus, even with a state-of-the-art CNN, a default or arbitrary choice may result from suboptimal to near-random classification results.

Learning rate scheduling effects: we used a fixed batch size of 32, and investigate two types of learning rate decay. Table 6 shows that, with SGD, the exponential type decay obtained slightly higher results when compared to the step type decay, highlighting those with pre-trained weights. For RAdam, exponential decay was better with Random weights, while Step Decay stands out using Pre Trained weights reaching $\approx 97\%$ accuracy with initial LR 0.0001.

Tabela 3. Classification accuracy (SVM) (%) of Cifar-10 dataset with a Resnet50 with SGD and RAdam optimizers from Random Initialization (Rnd) and Pre-trained (PT) with ImageNet.

SGD					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.5**	84.6 ±0.8	80.5 ±3.5	79.3 ±1.6	79.1 ±1.7
	0.1	78.8 ±0.2	71.6 ±2.1	66.4 ±1.3	62.3 ±2.9
	0.01	62.0 ±0.4	54.4 ±0.2	48.7 ±0.4	45.4 ±0.3
PT	**0.5**	80.7 ±0.7	77.3 ±3.1	79.8 ±1.2	76.3 ±2.4
	0.1	81.2 ±0.7	81.4 ±0.5	79.9 ±0.5	76.4 ±1.5
	0.01	80.2 ± 0.2	76.9 ± 0.4	73.9 ± 0.2	68.4 ± 0.3
RAdam					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.01**	57.3 ±6	66.0 ± 2.7	68.7 ±3.4	72.2 ±3.5
	0.001	84.1 ±0.7	84.4 ±0.4	82.3 ±0.4	78.1 ±0.3
	0.0001	64.6 ±0.6	59.6 ±0.9	53.2 ±0.7	48 ±0.4
PT	**0.01**	13.6 ± 7.0	33.1 ± 9.4	38.4 ±13.7	39.3 ±24.3
	0.001	63.0 ± 26.4	76.3 ±2.3	75.8 ± 3.2	65.0 ± 25.4
	0.0001	82.3 ± 0.5	82.3 ±0.3	81.3 ± 0.3	78.4 ± 0.3

Tabela 4. Classification accuracy (SVM) (%) of Cifar-10 dataset with a MobileNet with SGD and RAdam optimizers from Random Initialization (Rnd) and Pre-trained (PT) with ImageNet.

SGD					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.5**	65.2 ±15.5	67 ±1.5	64.7 ±1.2	60.4 ±0.8
	0.1	69.4 ±1.2	64 ±0.7	59.6 ±0.7	54.4 ±0.7
	0.01	59.8 ±0.8	55.1 ±0.7	50.1 ±0.4	45.3 ±0.7
PT	**0.5**	25.7 ±1.6	26.3 ±0.9	21.7 ±16.3	25.8 ±18.8
	0.1	81.7 ±0.4	80.6 ±0.9	70.3 ±18.2	67.3 ±17.4
	0.01	79.7 ±0.2	76.5 ±0.4	73 ±0.2	67.8 ±0.3
RAdam					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.01**	48.1 ±30.1	75.9 ±1.6	74.5 ±1.4	69.8 ±2.7
	0.001	75 ±0.8	74.1 ±0.7	68.3 ±1.3	59.2 ±2.5
	0.0001	54.7 ±0.4	47.1 ±0.4	39.3 ±0.7	31.7 ±1.1
PT	**0.01**	10 ±0	10 ±0	21.2 ±22.4	36.2 ±25.5
	0.001	83.4 ±0.6	83.3 ±0.3	82.5 ±0.1	81.8 ± 0.4
	0.0001	82.7 ±0.2	81.1 ±0.3	77.1 ±0.3	71.4 ±0.3

5.3. Guidelines

To summarize the results, we show color/bubble maps representing each combination of learning rate and batch size. Color indicate the average accuracy, while the size of the

Tabela 5. Classification accuracy (SVM) (%) of Flowers dataset with a MobileNet with SGD and RAdam optimizers from Random Initialization (Rnd) and Pre-trained (PT) with ImageNet.

SGD					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.5**	27.8 ±1.8	26.2 ±0.9	27.5 ±1.4	27.5 ±2.4
	0.1	38.9 ±26.1	27.4 ±1.5	34.8 ±14.2	28.2 ±1.7
	0.01	80.5 ±3.9	75.9 ±4.5	61.2 ±4	73.4 ±4
PT	**0.5**	45.1 ±12.8	48.3 ±14.6	33.7 ±6.7	34.1 ±5.1
	0.1	42.1 ±12.7	47.5 ±14.9	53.5 ±16.2	94.7 ±0.5
	0.01	93.9 ±1.1	91.8 ±1.9	90 ±1.3	86.2 ±0.4
RAdam					
Init.	**LR**	**Batchsize**			
		32	**64**	**128**	**256**
Rnd	**0.01**	45.3 ±25.7	66.7 ±7.6	53 ±9.5	69.7 ±5.9
	0.001	90.1 ±1.2	82.8 ±7.6	71.4 ±9.4	85.5 ±2.8
	0.0001	86.6 ±0.8	86.4 ±0.7	70.6 ±2.8	87.6 ±1.4
PT	**0.01**	33 ±4.7	34.1 ±3	33.6 ±9.3	89.9 ±2.8
	0.001	93.5 ±1.2	88.3 ±3.3	81.8 ±5.2	90.8 ±1.3
	0.0001	96 ±0.4	93.3 ±0.6	89.1 ±0.3	86.3 ±0.2

Tabela 6. Classification accuracy (SVM) (%) of Cifar-10 and Flowers dataset for a MobileNet architecture with Step Learning Rate Decay (SD) and Exponential Learning Rate Decay (ED) and optimizers SGD and RAdam from Random Initialization (Rnd) and Pre-trained (PT) with ImageNet.

SGD					
Init.	**LR**	**Cifar10**		**Flowers**	
		SD	**ED**	**SD**	**ED**
Rnd	**0.1**	62.7 ±0.9	63.3 ±1.1	28.4 ±3.4	28.7 ±3.1
	0.01	51.7 ±0.9	53 ±0.3	74.4 ±1.4	85.3 ±1.4
PT	**0.1**	81.6 ±0.6	81.9 ±0.7	35.3 ±5.5	38.7 ±6.6
	0.01	77.6 ±0.1	78.2 ±0.3	95.9 ±0.5	96.2 ±0.5
RAdam					
Rnd	**0.01**	60.9 ±25.5	58 ±24	94.1 ±0.3	94.1 ±0.8
	0.001	69.1 ±0.4	73.1 ±0.6	92.1 ±0.8	91.6 ±0.6
	0.0001	45.8 ±0.5	54.6 ±0.9	59.6 ±3.4	91.9 ±0.8
PT	**0.01**	10 ±0	10 ±0	34.9 ±4.6	31.8 ±0.4
	0.001	85.2 ±0.2	83.9 ±0.7	97.2 ±0.4	96.1 ±1.2
	0.0001	81 ±0.2	82.9 ±0.4	94.4 ±0.3	96.7 ±0.4

circle/bubble indicates the standard deviation. Thus, larger circles indicate scenarios that are sensitive to initialization. Absent circles represent near random results. The horizontal axis represents the batch size, while the vertical axis indicates the learning rate. Figure 3 shows the summary of SGD, Adam and RAdam in the first set of experiments. In Figure 4

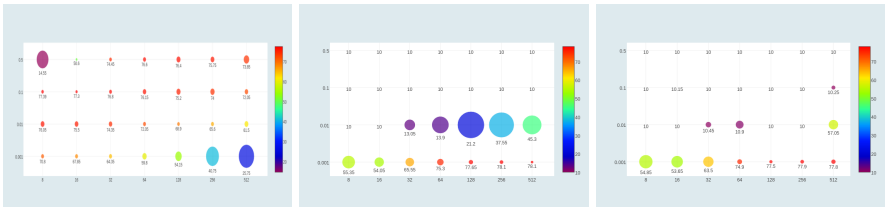
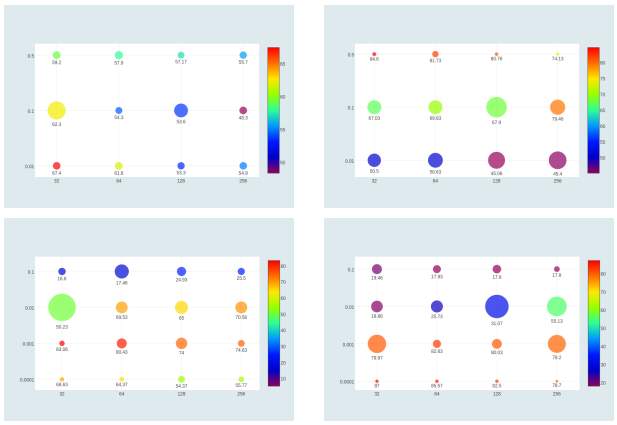


Figura 3. SGD (top), Adam (middle) and RAdam (bottom): summaries of first set of experiments considering mean and standard deviation of accuracies for different batchsize and learning rate combinations.



Random Pre Trained

Figura 4. SGD (top) RAdam (bottom) results with Resnet50 and MobileNet architectures and Cifar-10 and Flowers Recognition datasets. Random weight initialization (left) and pre-trained weights (right).

we compare random and pre-trained weights initialization.

We summarize the guidelines as follows:

— **Optimizer**: all optimizers are capable of achieving good results for both tasks, as long as we made appropriate choices. This corroborates previous studies that did not find a single best optimizer. However, Adam/RAdam seem to have advantages in transfer learning scenarios and for fine-grained datasets.

— **Learning rate starting value**: when learning from scratch we recommend exploring higher values for SGD (0.01-0.5) and lower values for Adam and RAdam (0.001-0.0001); values lower than the standard ones are recommended for pre-trained weights.

— **Learning rate scheduling**: both step and exponential decay are able to improve the learned representations with respect with no decay.

— **Batch size**: SGD tends to perform better for smaller batch sizes (8-32) and it is sensitive to larger batches, while Adam/Radam are better with larger batches (64-512). This choice must also take into account that smaller values require more running time, and larger values require more memory.

— **Random vs pre-trained weights**: SGD showed different behavior when used to optimize smaller and larger architectures. RAdam tend to be better for pre-trained weights, in this case we recommend investigating LR significantly smaller than the default

values.

6. Conclusion

Optimization choices matters when using deep networks for learning representations. As main conclusions: first, we show that SGD, Adam and RAdam are sensitive to the learning rate and batch size. Second, there is a non-obvious relationship between batch size and learning rate, which changes depending on and initialization (random or pre-trained). While larger batch size speeds-up training, it is paramount to adjusting the learning rate properly.

In general, the classical SGD works best with larger steps, while adaptive methods require smaller ones, in particular for fine-tuning. Although this is not a novel conclusion, we note that in deep learning frameworks, the default batch size is 32, while default LR is 0.01 for SGD and 0.001 for Adam/RAdam. Our results showed those to be suboptimal in various cases.

We offer guidelines for researchers and practitioners. By knowing the behaviour of optimization methods, one can significantly improve representation learning, without an exhaustive search of parameters. Future work may use our conclusions to investigate other phenomena such as overfitting, the role of augmentation methods and other learning tasks.

Referências

Abello, A. A. and Hirata, R. (2019). Optimizing super resolution for face recognition. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 194–201. IEEE.

Andrade, N., Faria, F. A., and Cappabianco, F. A. M. (2018). A practical review on medical image registration: from rigid to deep learning based approaches. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 463–470. IEEE.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Cauchy, M. A. (1847). Méthode générale pour la résolution de systèmes d'équations simul- tanées. *C. R. Acad. Sci Paris*, pages 25:536–538.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. (2019). On empirical comparisons of optimizers for deep learning. *CoRR*, abs/1910.05446.

Defazio, A., Bach, F. R., and Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *CoRR*, abs/1407.0202.

Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105.

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. Y. (2011). On optimization methods for deep learning. In *ICML*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems 31*, pages 6389–6399.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020). On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2015). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Mello, R. and Ponti, M. A. (2018). *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

Ponti, M. A., Ribeiro, L. S. F., Nazare, T. S., Bui, T., and Collomosse, J. (2017). Everything you wanted to know about deep learning for computer vision but were afraid to ask. In *2017 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*, pages 17–41. IEEE.

Ponti, M. A., Santos, F. P. d., Ribeiro, L. S. F., and Cavallari, G. B. (2021). Training deep networks from zero to hero: avoiding pitfalls and going beyond. *arXiv preprint arXiv:2109.02752*.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Rodrigues, L. F., Naldi, M. C., and Mari, J. F. (2017). Exploiting convolutional neural networks and preprocessing techniques for hep-2 cell classification in immunofluorescence images. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 170–177. IEEE.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Schmidt, M., Le Roux, N., and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1–2):83–112.

Schmidt, R. M., Schneider, F., and Hennig, P. (2020). Descending through a crowded valley - benchmarking deep learning optimizers.

Serpa, Y. R., Pires, L. A., and Rodrigues, M. A. F. (2019). Milestones and new frontiers in deep learning. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pages 22–35. IEEE.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Sivaprasad, P. T., Mai, F., Vogels, T., Jaggi, M., and Fleuret, F. (2019). On the tunability of optimizers in deep learning. *CoRR*, abs/1910.11758.

Sun, R. (2019). Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Tileman, T. and Hinton, G. (2012). Neural networks for machine learning. techinical report.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2018). The marginal value of adaptive gradient methods in machine learning.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., and Cong, J. (2015). Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170.