Adaptive Sharpness-Aware Minimization (ASAM) for Scale-Invariant Learning of Deep Neural Networks

Damien Gomez Donoso, Noam Ghenassia, Théau Vannier

Optimization for Machine Learning

INTRODUCTION

Machine learning has already proved itself in numerous domains. Nowadays, it is widely used all in many different applications. Deep learning in particular showed a remarkable capacity of adaptation. If well trained, a neural network is able to solve most problems thanks to its universal function approximator property. Moreover, it showed a great ability to generalize to unseen data. All these features make it a precious tool to tackle most artificial intelligence problems. Nevertheless, generalizing beyond the training set remains a big challenge. For instance, a classifier can underperformed on a slightly noised input. This is especially true when the training on insufficient data to truthfully approximate the real distribution. In this case, a minimum of the loss function can be very data-dependent and thus not represent a real minimum of the actual underlying distribution we want to approach. This motivated the idea of Sharpness Aware Minimization (SAM). The main focus of our work is the added functionality provided in ASAM [KKPC21] that adresses fundamental limitations of the SAM algorithm (the reader may refer to appendix A for a brief summary of the principles of SAM).

A. Positive scaling transformations

Sharpness aware minimization allows a great progress in terms of generalization of the loss function. However, minimizing the loss of a neural network can yield many combinations of weights that that give rise to the same neural network function. An easy example would be a neural network with one input neuron, one output neuron and one hidden layer containing a unique neuron. By minimizing the loss function we can obtain the optimal weights w_1^* and w_2^*. With the ReLU activation function, one can obtain the exact same outputs using the weights $w_1^{'} = \lambda \cdot w_1^*$ and $w_2^{'} = \frac{1}{\lambda} \cdot w_2^*$.

B. Adaptive Sharpness-Aware Minimization : ASAM

ASAM is an extension of the SAM minimizer that tries to take into account this equivalence of parameter vectors in its minimization process. To do so, it applies a transformation on the sharpness term such that it stays constant for parameter vectors that support the same function ([KKPC21]). The equivalence of parameter vectors is explained in more details in the next section. One can define T_w^{-1}, a normalization transformation of w such that given a weight w, $T_{Aw}^{-1} A = T_w^{-1}$ for any invertible scaling operator A on $\mathbb{R}^k$ which does not change the loss function. Using this normalization transformation, we can define the adaptive sharpness as,

$$\max_{||T_w^{-1}\epsilon||_p \leq \rho} L_S(w+\epsilon) - L_S(w) = \max_{||T_{Aw}^{-1}\epsilon||_p \leq \rho} L_S(Aw+\epsilon) - L_S(Aw),$$

where $1 \leq p \leq \infty$

The equality of this equation comes from the fact that A is an invertible scaling operator that does not change the loss function.

This introduction gives us the tools to understand the main idea of ASAM and SAM. In this project, we want to test the principle of translation invariances that are used in the ASAM method. Then, we want to test these methods on real datasets to observe their impacts.

More precisely, in this report, we will focus on the following three main points :

- First, we will illustrate how ASAM is independent on positive scaling with some experiences on a toy model.
- Then we will show the generalization efficiency of ASAM and SAM compared to ADAM when training these methods on hand-crafted toy datasets.
- Finally, we will show its efficiency on a larger and more interesting dataset : cifar10.

I. PARAMETER VECTORS EQUIVALENCE

In this section, we investigate how ASAM works under the hood. The objective of ASAM is to define a sharpness measurement that is independent on positive scaling. Given a network architecture, the parameter space $\mathcal{W}$ is the set of all possible weights and biases combinations, and f_w is the function supported by the network with the parameters $w \in \mathcal{W}$. Since each parameter has a real value, if the total number of parameters is d, the parameter space is isomorphic to $\mathbb{R}^d$. Let us assume that the total number of neurons in the hidden layers is p, then a positive scaling transformation is defined by a vector $\tau = (\tau_i)_{i \in \{1,...,p\}}$, with $\tau_i > 0 \ \forall \ i \in \{1, ..., p\}$. We can apply these transformations to obtain a new parameters vector $w' = \tau w$: for each neuron p_i, we multiply all its incoming weights by τ_i and divide all of its outgoing weights by the same value. Therefore, we can define an equivalence relation on $\mathcal{W}$:

$$w \sim w' \iff \exists\, \tau \in \mathbb{R}^p_{>0} \colon w' = \tau w$$

The equivalence classes of the relation are submanifolds of the quadrants of $\mathcal{W}$ with the property that all the networks with parameter

vectors in the same class support the same function : for each input x in the input space, $w' = \tau w \Rightarrow f_w(x) = f_{w'}(x)$. Crucially, while the function of the networks (and therefore the loss value) remains the same for all parameter vectors in the equivalence class, the sharpness of the loss landscape can vary arbitrarily. The aim of ASAM is therefore to propose a sharpness related metric that does not depend on the representative of the equivalence class.

The proposed transformation introduced in section -B is $T_w = \text{diag}(|w_1|, ..., |w_d|)$: at each ascent step, the ϵ vector is multiplied in each dimension by the absolute value of the corresponding weight. Importantly, this rescaling of the ϵ vector only happens in the dimensions corresponding to weights, but not in the dimensions corresponding to biases.

In order to illustrate the affect of ASAM, we perform the following experiments : we randomly initialize a neural network, and perform 50 random positive scaling transformations (i.e., transformations parametrized by τ vectors with entries uniformly distributed in the interval $[0.5, 1.5]$) and plot the output (logits) of the network for some randomly selected input points (see fig 2). As we can observe, the positive scaling transformations have no effect on the network's output.

We then repeat the experiment twice, and plot the SAM and ASAM losses. We can observe on figure 3 that the transformations have a dramatic effect on the SAM loss value. On the other hand, figure 4 shows that the ASAM loss remains practically constant during the whole process.

An additional point can be made here. A study of ASAM's implementation reveals that it only rescales the weights of the network, but not the biases. This was confirmed when we applied the positive scaling transformations on both the weights and the biases. We could then observe that the ASAM loss also varied dramatically. This indicates that the the current implementation of ASAM only considers positive scaling transformations that leave the biases unchanged. Some possible future work can therefore try to find a sharpness related measure that is invariant to both the weights and biases of the network.

II. Training on a toy dataset

In order to illustrate the benefits of SAM and ASAM, we first compare them on a simple dataset. We want to show that ASAM and SAM generalize better than ADAM. To do so, we construct a toy 2D dataset drawn from a known distribution whose variance can be controlled. Some examples are shown on table 6. To test the generalization ability of ADAM, ASAM and SAM, we train a model with each optimizer on 5 datasets with varying variances. Once trained, we test these neural networks on datasets with different variances and we compare the test errors.
We use the library pytorch to create and train the neural networks. The three neural networks have the same loss function (i.e. cross entropy) and the same structure B.

The difference between these neural networks are the trainings. The neural network used with ADAM uses only the ADAM optimizer. The two others use ADAM optimizer and ASAM/SAM minimizer both with $\rho = 0.5$ and $\eta = 0.01$. These minimizers can be found on Samsung's Github repository. For the three methods we also use a learning rate scheduler, we use the CosineAnnealingLR form the library pytorch. For the reporductibility of this experiement, we set the seed to 0.

We then test the trained networks on test datasets with increasing variances. The results are shown on fig 5a, 5b, 5c, 5d and 5e.
We can observe that networks trained with ASAM and SAM generalize better than training with ADAM because, for test datasets with high variances, the blue curve is always above the green and the yellow curves.
interestingly, the curves behave in the same way for the different variances of the test datasets (except for too noisy datasets). If we increase the variance of the test datasets (starting from 0), ADAM first gives the lowest test errors. But when the variances of the test datasets reach values around the variance of the train dataset, ASAM becomes the best method because it gives the lowest test errors. If we keep going to increase the variance of the test datasets, there is a point where SAM starts to give the lowest test errors and keeps giving the lowest errors for all the variances after this point.
For example, if we look at the result shown on figure 5c (train dataset with $\sigma^2 = 15$), ASAM method gives the lowest test errors for test datasets with variances from 5 to 40 and, after $\sigma^2 = 40$, SAM method gives the lowest test errors. For

variances below $\sigma^2 = 5$, ADAM gives the lowest test errors.
If we take a train dataset with a high variance (figure 5e), i.e. a noisy dataset, the improvement in test errors between ASAM and ADAM is less clear because there isn't a big gab between their curves. Nevertheless, SAM keeps giving the lowest test error for test datasets with high variances.

To summarize, we can conclude that SAM and ASAM generalize better than ADAM. More precisely, ASAM generalizes better than SAM for datasets with variances that are in a range of values that can be large, around the variance of the train dataset. For test datasets with variances larger that this range of values, SAM generalizes better.

III. Training on cifar10

In this section we want to show ASAM's efficiency in a more realistic setting. To do so we attempt to reproduce the results obtained by SAMSUNG in their paper of ASAM [KKPC21]. We train a ResNet model : wrn28_10 ([ZK16]) on cifar10 using respecitvely the ADAM optimizer, SAM and ASAM minimizers. ADAM is a widely used and very efficient optimizer, that is why we took it as a reference for the comparison. Because of the computational cost of training ResNet, we could not do more than 50 epochs, which still gives us quite interesting results, which are presented in the following table where we can see that SAM and ASAM are clearly above ADAM in terms of performance. Regarding SAM and ASAM, it is hard to tell whether ASAM did significantly improve the performance compare to SAM in this specific case as we did only one simulation and "only" 50 epochs. Our results are consistent with Samsung's, we could reproduce their results. In their case, they did several simulations up to 200 epochs and could confirm a significant, yet not so big, improvement between SAM and ASAM. But still SAM is already very good on cifar10, maybe a greater difference could be seen on another dataset harder to train for SAM. A final interesting remark is that we also tried to train a very simple model on cifar10 for 200 epochs. Of course the model is too simple to handle cifar10, but still we were surprised to see that in this case sharpness aware optimizers did not improve the performances. It should also be noted that while the SAM and ASAM optimizers did yield significantly better performances, they also took longer to use. We trained our models on a RTX 3060 GPU, and the average time per epoch were 4:30 minutes with ADAM, 6:30 minutes with SAM and 9:00 minutes with ASAM. Therefore, it is possible that if ADAM was given as much *time* rather than as many *epochs*, it would have yielded similar performances as the sharpness aware optimizers. It is also possible, however, that ADAM would converge before reaching higher performances.

	ADAM	SAM	ASAM
Accuracy(%)	92.5	96.19	96.44

Table I: Best test accuracy of ResNet for different optimizers after 50 epochs

In addition, we also plotted the accuracies for the different optimizers to see how it behaves will training. An important remark is that the plots of SAM and ASAM have a very small gap between training and testing accuracies (fig.7b and fig.7c) compared to ADAM (fig.7a). This confirms our intuition that ASAM helps to generalize the model and therefore is less subject to dataset variations. In this case we can see Sharpness minimization as a kind of regularization which helps generalizing the model.

Conclusion

In this work, we reviewed the sharpness aware optimizers. In particular, we focused on ASAM, an optimizer that infers the generalization ability from a sharpness related measure of the loss landscape. This measure, called adaptive sharpness, is independent on positive scaling transformations. We empirically demonstrated that the adaptive sharpness value remains practically unchanged under these positive scaling transformations. We then illustrated the benefits of using the sharpness aware optimizers on a toy example that allowed us to derive a criterion for choosing the best optimizer for a given usecase (among ADAM, SAM and ASAM). Finally, we observed that on a large model, when the number of training epochs is limited, sharpness aware optimizers yield significantly better performances.

References

[FKMN20] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2020.

[KKPC21] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks, 2021.

[KMN+16] 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, 2016.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

APPENDIX A
THE SAM ALGORITHM

A. Generalizability and loss landscape

To train robust models, we need a way of avoiding the optimization process to end in a minimum that does not generalize. This is usually the case of sharper minima. Indeed, one should keep in mind that due to finite size effects, the training loss landscape is not exactly equal to that of the actual distribution. As a result, sharper minima of the training loss function yield higher test loss values. This is easily visualized in Fig 1 : a small shift between the training loss landscape and the expected loss over the whole distribution of the datapoints yields a higher increase of the loss value when the minimum is sharp, even if the loss value in both minima is equal. This motivates the idea of constraining the loss function by an additional sharpness term.

B. Sharpness Awareness Minimization (SAM)

Instead of only looking for the point that gives the lowest loss value, the sharpness term helps to find a minimum w^* which lies in a neighborhood that smoothly goes toward this minimum. The key idea behind doing so is that if we take a new sample of datapoints, the "smooth minimum" previously found should not move too much. This way we obtain a more robust minimum which better generalizes and is not specific to the training set. In the paper of SAM, they could show its robustness to label noise on par with that provided by state-of-the-art procedures that specifically target learning with noisy labels. [FKMN20] More formally, the motivation for SAM comes from the following theorem :
For any $\rho > 0$ (radius of a L_2-ball), with high probability over training set S generated from distribution D,

$$L_D(w) \leq L_S(w) + [\max_{||\epsilon||^2 \leq \rho} L_S(w+\epsilon) - L_S(w)] + h(\frac{||w||_2^2}{2})$$

where L_D is the population loss, L_S is the training loss over the training set S and $h : \mathbb{R}_+ \to \mathbb{R}_+$ is a strictly increasing function (under some technical conditions on $L_D(w)$).
This theorem tells us that the true is upper bounded by the loss of a sample S, plus the sharpness term, plus a quadratic term. It leads us to a "min-max" problem :

$$\min_w \max_{||\epsilon||_p \leq \rho} L_S(w+\epsilon) + \lambda ||w||_2^2$$

where $\rho \geq 0$ is a hyperparameter and $p \in [1, \infty]$ (generalized norm). We see from this formula that we want to find the optimal w^* that minimizes the loss not only in w^* but also in the largest possible neighborhood $\{w \in \mathbb{R} \mid ||w - w^*||_p \leq \rho\}$. Essentially, at each step the loss function is evaluated not at the current point in the parameter space, but rather at the point that maximizes the loss within a ball of radius ρ. In practice, SAM evaluates the loss at the point $w + \epsilon$ that maximizes the first order approximation of the loss within $B(w, \epsilon)$: Hence, ϵ is simply the rescaled gradient $\frac{\epsilon}{||\nabla L_S||} \cdot \nabla L_S$.

APPENDIX B
FIGURES

Order	Type	Name
1	input layer	nn.Linear(2, 10)
2	hidden layer	nn.Linear(10, 10)
3	activation function	nn.Relu()
4	hidden layer	nn.Linear(10, 10)
5	activation function	nn.Relu()
6	hidden layer	nn.Linear(10, 10)
7	activation function	nn.Relu()
8	output layer	nn.Linear(10, 2)

Table II: Structure of the neural networks

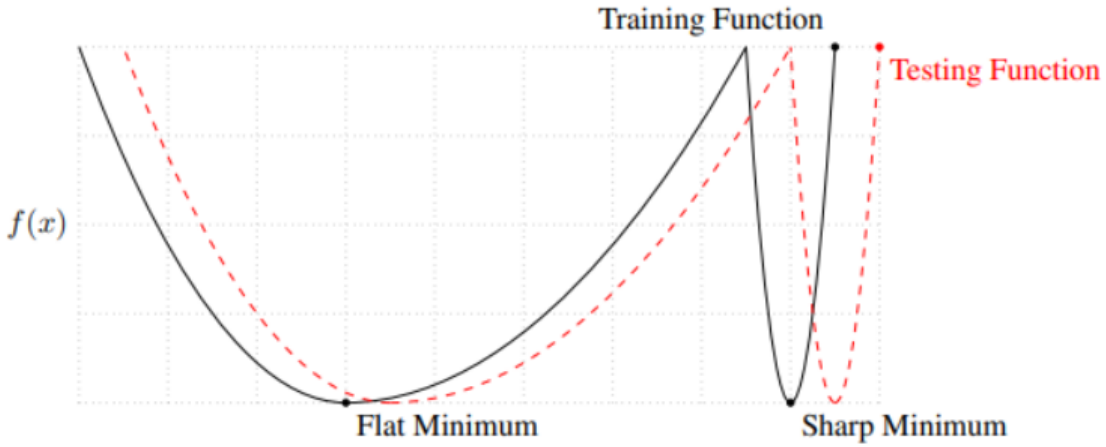


Figure 1: Flatter minima generalize better than sharper ones. This figure is taken from [KMN+16]

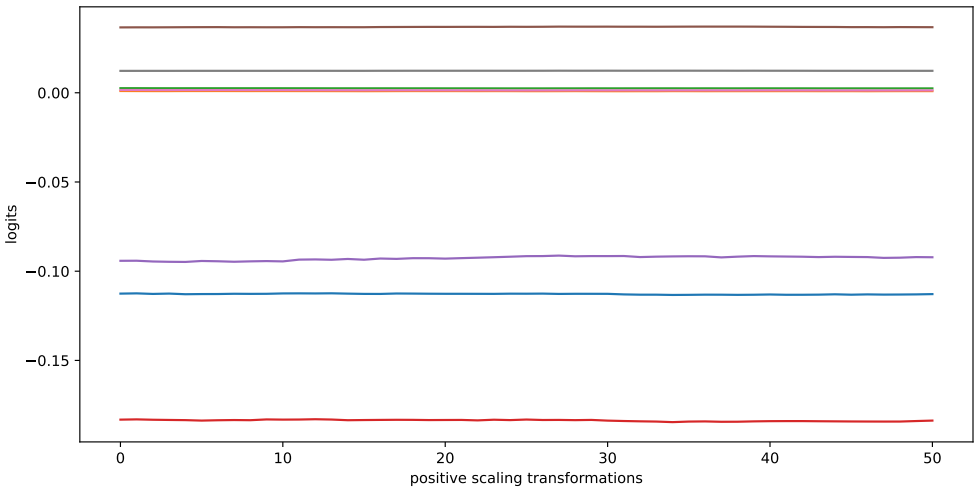


Figure 2: Output of the network for a few datapoints throughout positive scaling transformations.

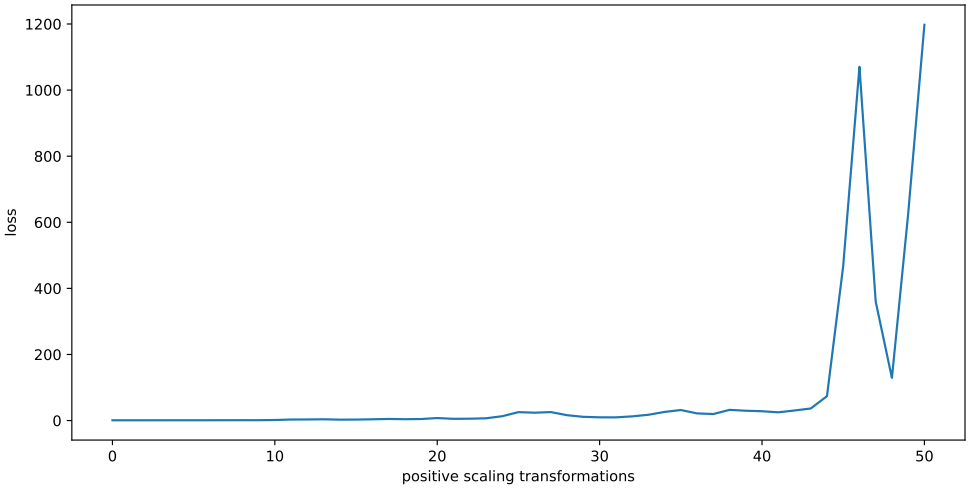


Figure 3: SAM loss of a network throughout positive scaling transformations.

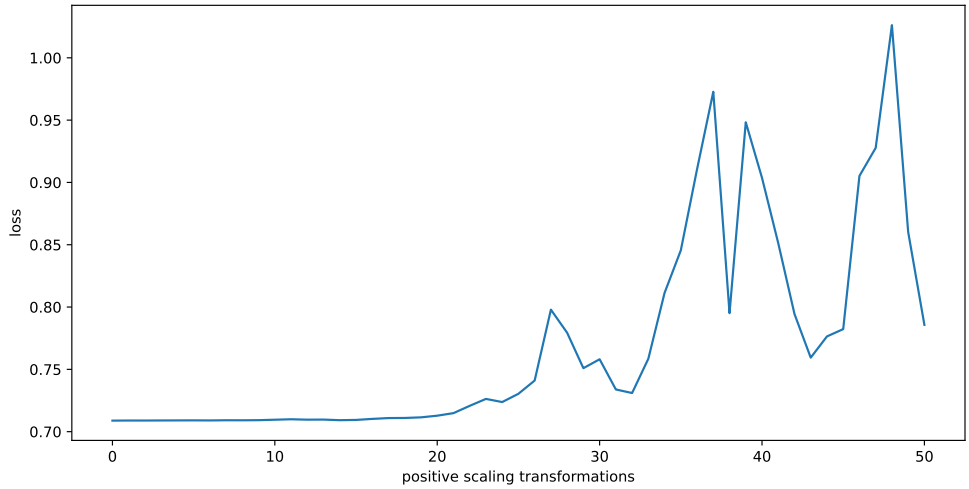
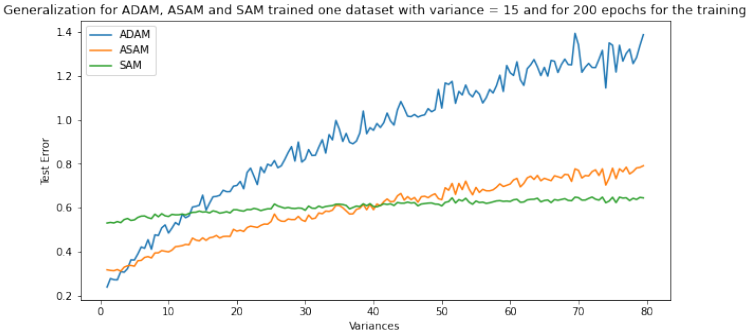


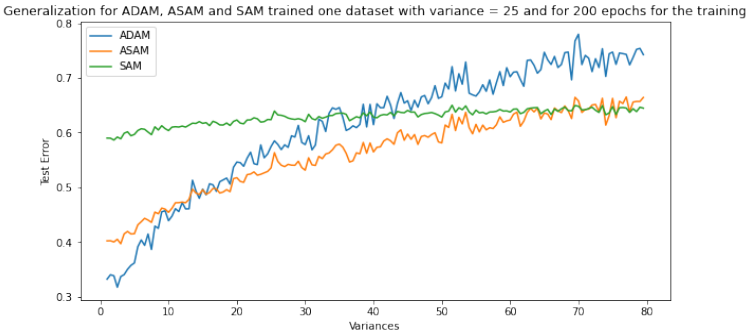
Figure 4: ASAM loss of a network throughout positive scaling transformations.

(a) Results for training dataset ($\sigma^2 = 1$).

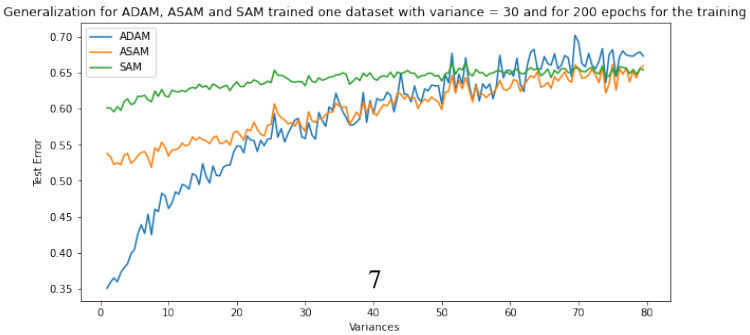
(b) Results for training dataset ($\sigma^2 = 5$).



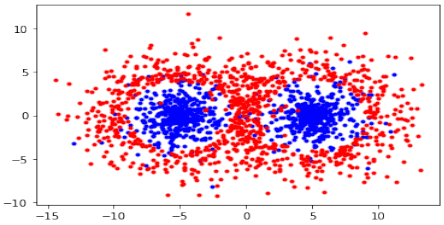
(c) Results for training dataset ($\sigma^2 = 15$).



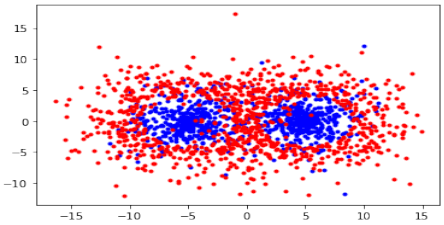
(d) Results for training dataset ($\sigma^2 = 25$).



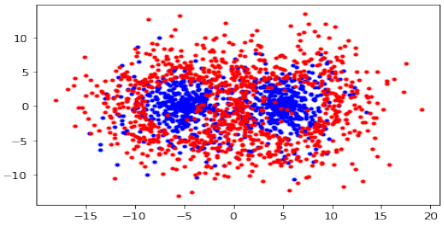
(e) Results for training dataset ($\sigma^2 = 30$).



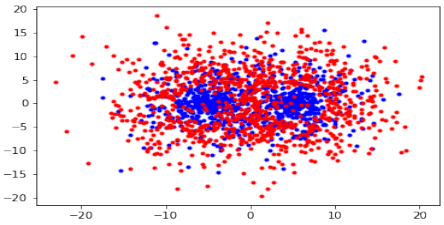
(a) Variance = 5



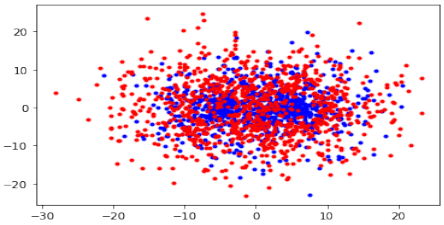
(b) Variance = 10



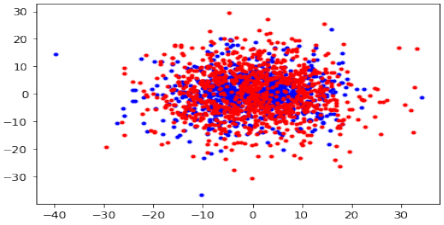
(c) Variance = 15



(d) Variance = 30

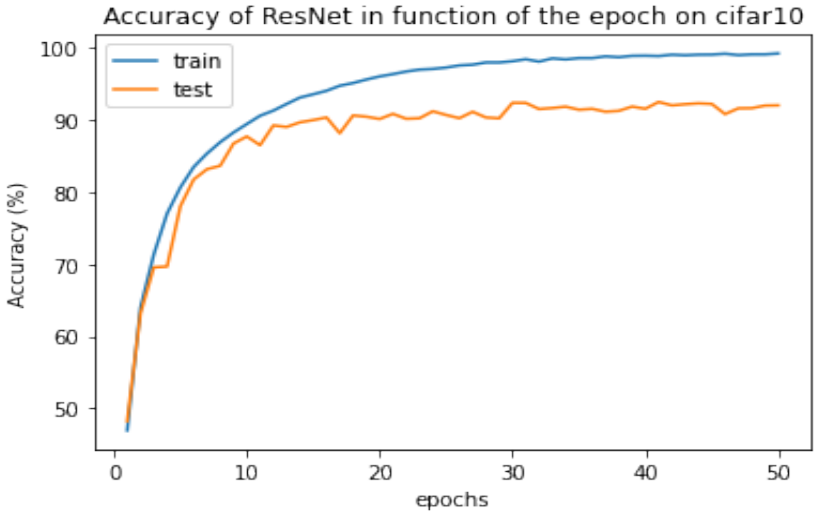


(e) Variance = 50

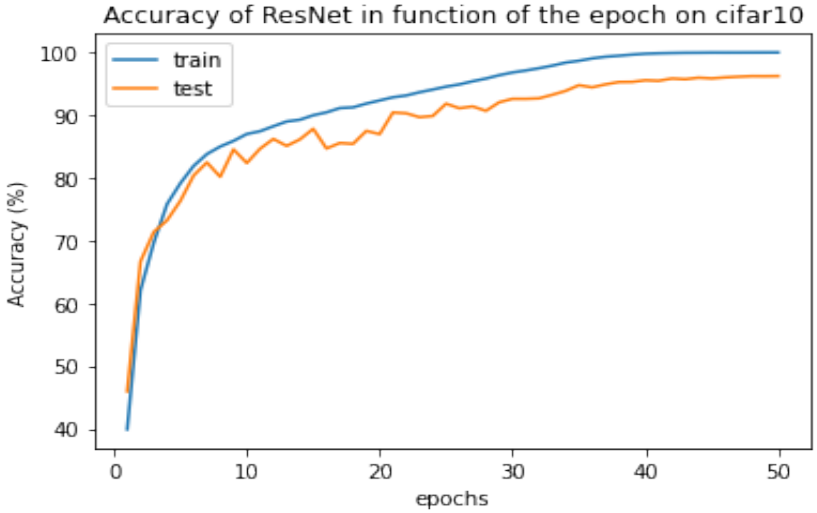


(f) Variance = 80

Figure 6: Example of toy datasets with different variances and with 2000 data points



(a) ADAM



(b) SAM

(c) ASAM

Figure 7: Accuracies of ResNet in function of the epoch on cifar10 with different optimizers