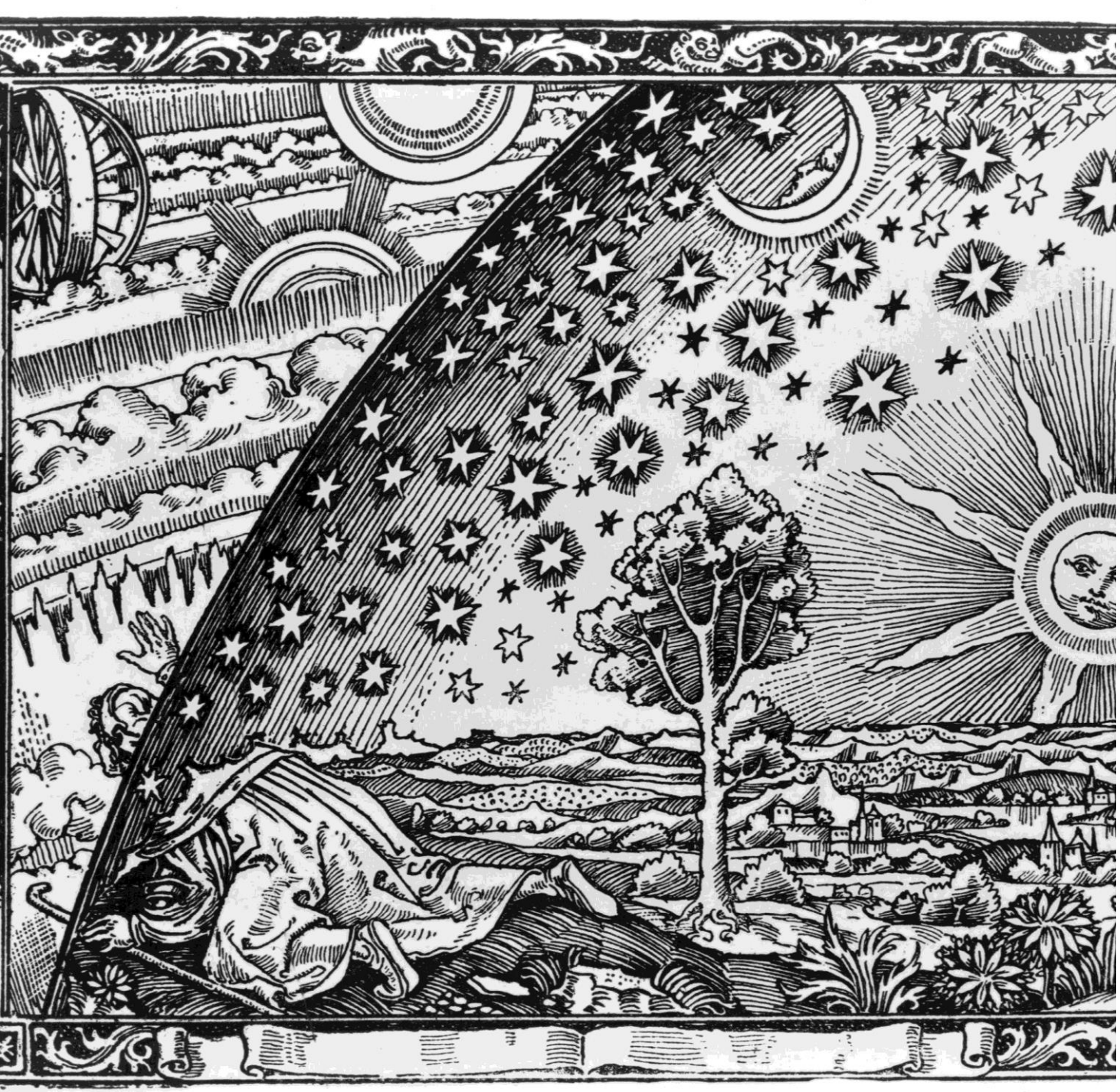
Generalization Boosts

Mathias Neitzel
Deep Learning Research Kitchen

- Introduction to Generalization
  - Generalization and Regularization
  - Flat Minima
  - Generalization in Deep Learning
  - Implicit Biases
- Tricks for Generalization
  - Weight Decay
  - Sharpness Aware Minimization
- Experiment
  - Dataset
  - Model
- Results


Introduction to
Generalization



Generalization

- Generalization refers to the ability of a model to adapt to new, previously unseen data that was not used to create or train the model.

- “Flammarion Engraving”

  Unknown Artist
  First published 1888

Measuring Generalization

- There is a wide range of methods (that try) to measure the ability of models to generalize well:
  - Vapnik-Chervonenkis Dimension
  - Rademacher Complexity
  - Uniform stability
- usually referred to as complexity measures.

Measuring Generalization

- Supervised Learning
  - Train-Test Split
  - Training on the Train Data
  - Testing on the test data

- Unsupervised Learning
  - Complicated Question
  - Example: “Generalization in Unsupervised Learning” (2015), Abou-Moustafa and Schuurmans

Regularization (in Theory)

- Methods that improve the ability of a model to get a low-test error are referred to as regularization methods and can be defined as:

  “Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”
  ~”Deep Learning” by Ian Goodfellow

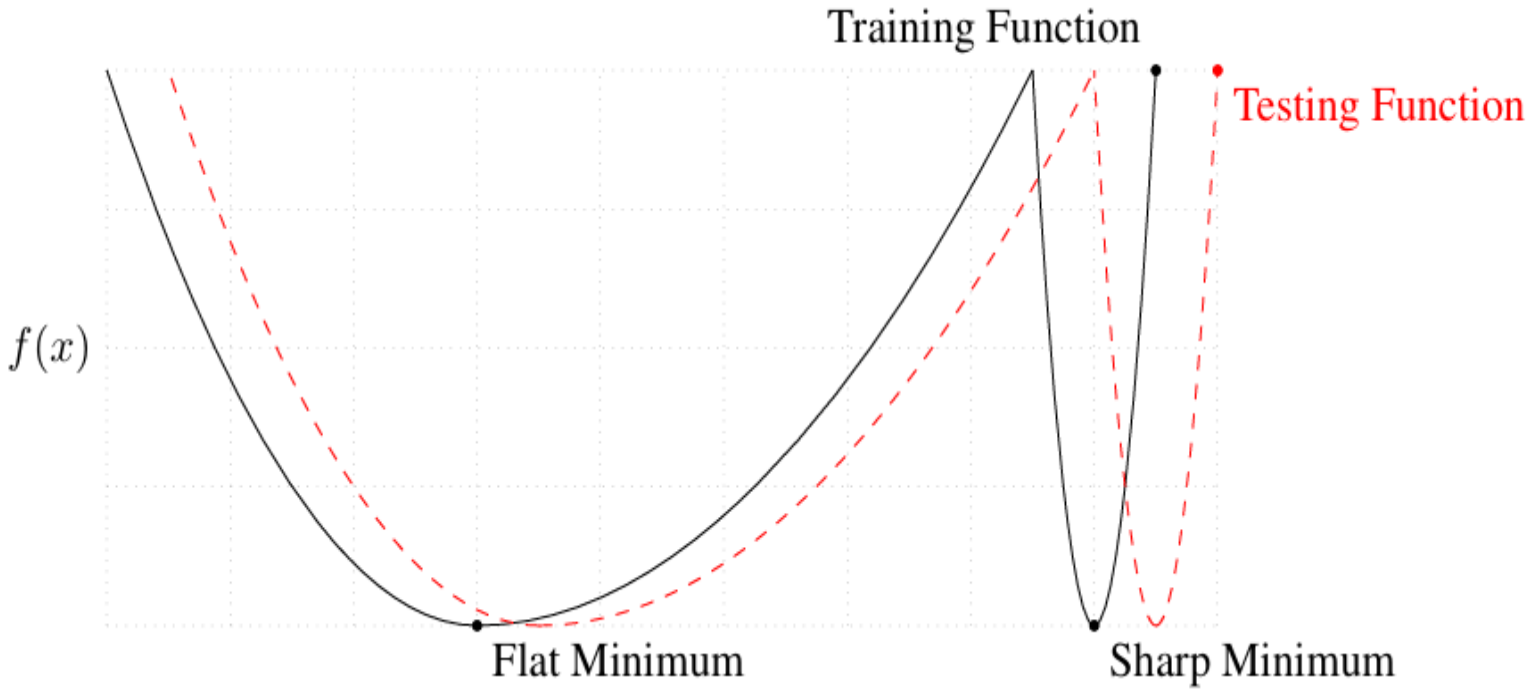
Regularization (in Practice)

- Methods that tend to simplify our model (e.g. reducing the magnitude of some parameters) are often referred to as being “regularizing” and vice versa.
- A regularization method can also be intended to have other positive effects (e.g. reduce training loss).
- Language inconsistency.

Flat Minima

- A „Flat Minima“ is a region in the loss landscape where the loss is low and remains relatively constant.

- “Flatter” minima
  ➔ “Better” generalization



Source: “ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA”

Generalization in Deep Learning

- Modern Deep Learning Models are „overparameterized“ they have the capacity to memorize any training data.
- Intuition would tell us that a model like this is prone to overfit and therefore would‘nt generalize well.
- It‘s the opposite! Deep Learning Models tend to generalize really well. We just dont know why exactly.

UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

- Deep neural networks easily fit random labels.

- “Traditional measures of model complexity struggle to explain the generalization ability of large artificial neural networks”

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Training on Cifar-10

Fantastic Generalization Measures and Where to Find Them

- Extensive testing of more then 40 complexity Measures on over 10.000 CNN's.
- Failure of many complexity Measures if tested on a diverse set of models.
- Still some promising candidates were found (especially PAC-Bayesian bounds).

On the Implicit Bias in Deep-Learning Algorithms

- “…when using an appropriate step size we can rule out stable convergence to certain (sharp) minima, and thus encourage convergence to flat minima,…”
- Deep Learning models with Gradient based Methods (and increased step size and smaller batches) converge towards flat minima.

Do optimizers produce an implicit bias in the solution at convergence?

- The noise in SGD helps SGD to leave local and sharp minima and thereby leads to SGD being implicitly biased towards flatter and global minimas.
- The same can not be said for Adam which is less locally unstable. (1)

- Tho in general we can always cause gradient noise by using (mini) batches. (2)

1: “Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning”
2:”Shape Matters: Understanding the Implicit Bias of the Noise Covariance”


Tricks for
Generalization

Weight Decay(WD)

- Simple Definition

- $\theta_{t+1} \leftarrow \theta_t - \eta \left(\frac{\partial L}{\partial \theta - t}\right) - \eta\lambda\theta_t$

- WD changes the update rule by decaying the weights depending on there own value, the learning rate η and a constant factor λ .

- A sort of „gravitational pull“ attracting weights toward zero.

Weight Decay VS L^2 Regularzation

- Weight Decay is not L^2 Regularzation.
- L^2 Regularzation applies a weight-penality on the Loss Function L
- $L_{reg}(\theta) = L(\theta) + \lambda \,||\, \theta \,||_2$
- Literature is highly inconsistent on this one.
  Even “Deep Learning” by Ian Goodfellow uses the terms interchangeable.

Weight Decay VS Normalization

- WD has no actual regularizing effect (in models with Layer/Weight normalization) and instead improves generalization in different ways.(1) (2)

- By decreasing the weights, weight decay increases the effective learning rate, contrary to the intuition that regularization results in a more stable model.(1)

- Smaller weights ➔ bigger impact of the lr

(1) “L2 Regularization versus Batch and Weight Normalization”
(2) “Three Mechanisms of Weight Decay Regularization”

Sharpness Aware Minimization (SAM)

- SAM is an Algorithm first mentioned in 2020
- SAM is designed to minimize loss and sharpness.

SAM Pseudocode

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

The math derivation why SAM works is hard.

The pseudo code is unintuitive.

The takeaway from the pseudo are:

-SAM has the hyperparameter p
-SAM is build on top of an optimizer

SAM Schematic

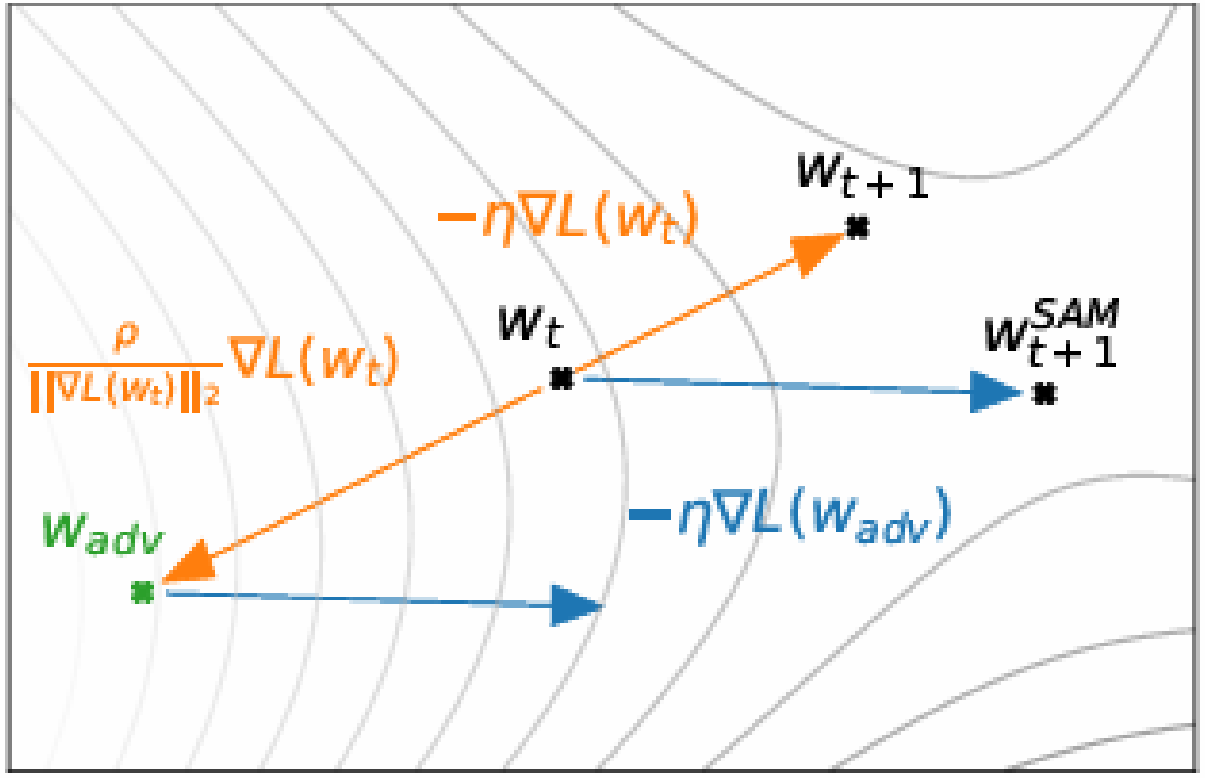


Figure 2: Schematic of the SAM parameter update.

Effectiveness of WD

Table 1: Classification results on CIFAR-10 and CIFAR-100. **B** denotes BN while **D** denotes data augmentation, including horizontal flip and random crop. **WD** denotes weight decay regularization. Weight decay regularization improves the generalization consistently. Interestingly, we observe that weight decay gives an especially strong performance boost to the K-FAC optimizer when BN is turned off.

Dataset	Network	B	D	SGD		ADAM		K-FAC-F		K-FAC-G	
					WD		WD		WD		WD
CIFAR-10	VGG16			83.20	84.87	83.16	84.12	85.58	89.60	83.85	**89.81**
		✓		86.99	88.85	88.45	88.72	87.97	89.02	88.17	**89.77**
		✓	✓	91.71	93.39	92.89	93.62	93.12	**93.90**	93.19	93.80
CIFAR-10	ResNet32			85.47	86.63	84.43	87.54	86.82	90.22	85.24	**90.64**
		✓		86.13	90.65	89.46	90.61	89.78	**91.24**	89.94	90.91
		✓	✓	92.95	95.14	93.63	94.66	93.80	**95.35**	93.44	95.04
CIFAR-100	VGG16	✓	✓	68.42	73.31	69.88	**74.22**	71.05	73.36	67.46	73.57
CIFAR-100	ResNet32	✓	✓	73.61	77.73	73.60	77.40	74.49	78.01	73.70	**78.02**

Source: “THREE MECHANISMS OF WEIGHT DECAY REGULARIZATION”

Effectiveness of SAM

		CIFAR-10		CIFAR-100	
Model	Augmentation	SAM	SGD	SAM	SGD
WRN-28-10 (200 epochs)	Basic	**2.7**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.5**$_{\pm 0.2}$	18.8$_{\pm 0.2}$
WRN-28-10 (200 epochs)	Cutout	**2.3**$_{\pm 0.1}$	2.6$_{\pm 0.1}$	**14.9**$_{\pm 0.2}$	16.9$_{\pm 0.1}$
WRN-28-10 (200 epochs)	AA	**2.1**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**13.6**$_{\pm 0.2}$	15.8$_{\pm 0.2}$
WRN-28-10 (1800 epochs)	Basic	**2.4**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.3**$_{\pm 0.2}$	19.1$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	Cutout	**2.1**$_{\pm 0.1}$	2.7$_{\pm 0.1}$	**14.0**$_{\pm 0.1}$	17.4$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	AA	**1.6**$_{\pm 0.1}$	2.2$_{\pm <0.1}$	**12.8**$_{\pm 0.2}$	16.1$_{\pm 0.2}$
Shake-Shake (26 2x96d)	Basic	**2.3**$_{\pm <0.1}$	2.7$_{\pm 0.1}$	**15.1**$_{\pm 0.1}$	17.0$_{\pm 0.1}$
Shake-Shake (26 2x96d)	Cutout	**2.0**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**14.2**$_{\pm 0.2}$	15.7$_{\pm 0.2}$
Shake-Shake (26 2x96d)	AA	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**12.8**$_{\pm 0.1}$	14.1$_{\pm 0.2}$
PyramidNet	Basic	**2.7**$_{\pm 0.1}$	4.0$_{\pm 0.1}$	**14.6**$_{\pm 0.4}$	19.7$_{\pm 0.3}$
PyramidNet	Cutout	**1.9**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**12.6**$_{\pm 0.2}$	16.4$_{\pm 0.1}$
PyramidNet	AA	**1.6**$_{\pm 0.1}$	1.9$_{\pm 0.1}$	**11.6**$_{\pm 0.1}$	14.6$_{\pm 0.1}$
PyramidNet+ShakeDrop	Basic	**2.1**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**13.3**$_{\pm 0.2}$	14.5$_{\pm 0.1}$
PyramidNet+ShakeDrop	Cutout	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**11.3**$_{\pm 0.1}$	11.8$_{\pm 0.2}$
PyramidNet+ShakeDrop	AA	**1.4**$_{\pm <0.1}$	1.6$_{\pm <0.1}$	**10.3**$_{\pm 0.1}$	10.6$_{\pm 0.1}$

Table 1: Results for SAM on state-of-the-art models on CIFAR-{10, 100} (WRN = WideResNet; AA = AutoAugment; SGD is the standard non-SAM procedure used to train these models).

Source: “SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION”

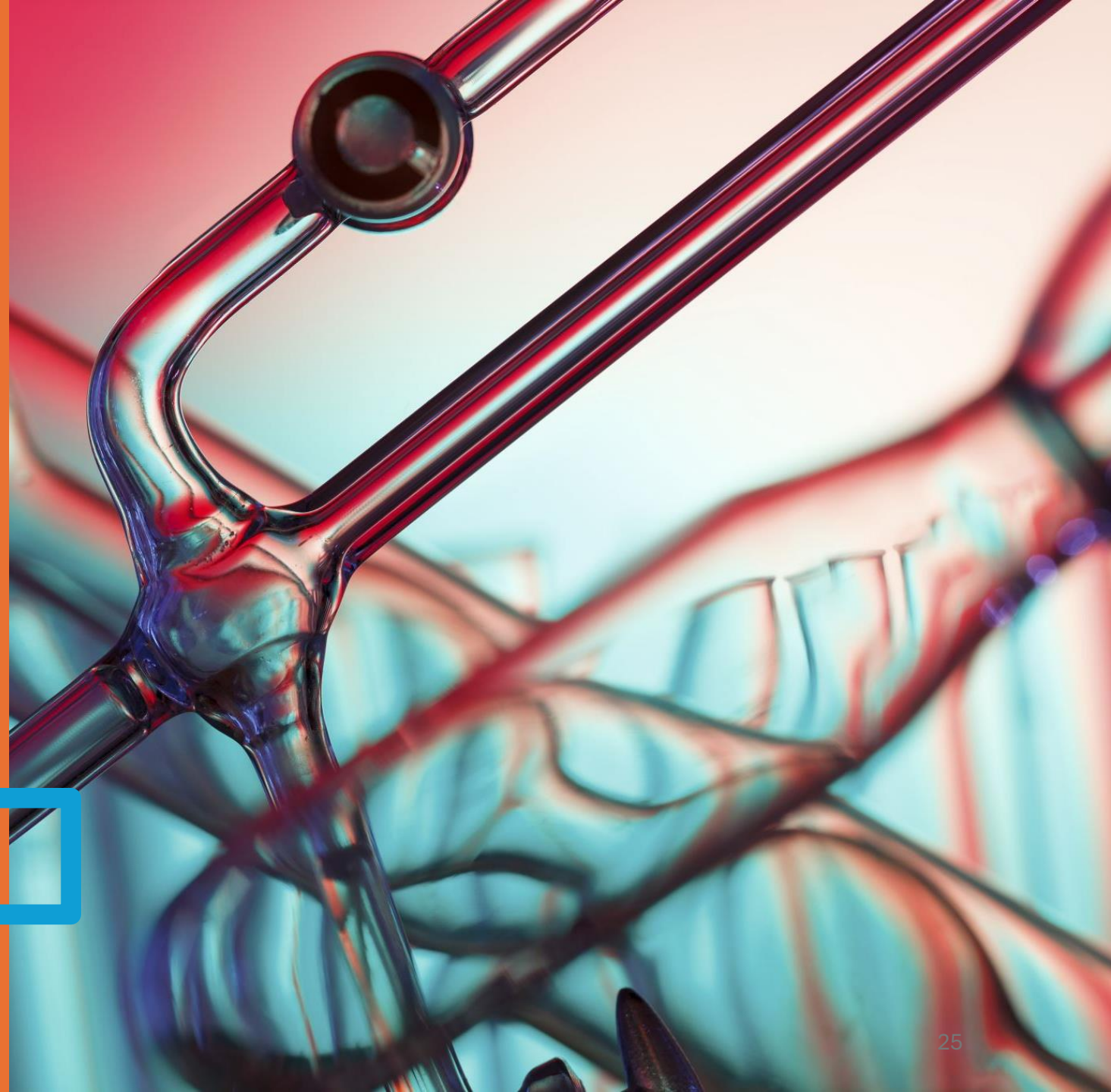
Effectiveness of SAM

Model	Epoch	SAM		Standard Training (No SAM)	
		Top-1	Top-5	Top-1	Top-5
ResNet-50	100	**22.5**$_{\pm 0.1}$	6.28$_{\pm 0.08}$	22.9$_{\pm 0.1}$	6.62$_{\pm 0.11}$
	200	**21.4**$_{\pm 0.1}$	5.82$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.37$_{\pm 0.04}$
	400	**20.9**$_{\pm 0.1}$	5.51$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.40$_{\pm 0.06}$
ResNet-101	100	**20.2**$_{\pm 0.1}$	5.12$_{\pm 0.03}$	21.2$_{\pm 0.1}$	5.66$_{\pm 0.05}$
	200	**19.4**$_{\pm 0.1}$	4.76$_{\pm 0.03}$	20.9$_{\pm 0.1}$	5.66$_{\pm 0.04}$
	400	**19.0**$_{\pm <0.01}$	4.65$_{\pm 0.05}$	22.3$_{\pm 0.1}$	6.41$_{\pm 0.06}$
ResNet-152	100	**19.2**$_{\pm <0.01}$	4.69$_{\pm 0.04}$	20.4$_{\pm <0.0}$	5.39$_{\pm 0.06}$
	200	**18.5**$_{\pm 0.1}$	4.37$_{\pm 0.03}$	20.3$_{\pm 0.2}$	5.39$_{\pm 0.07}$
	400	**18.4**$_{\pm <0.01}$	4.35$_{\pm 0.04}$	20.9$_{\pm <0.0}$	5.84$_{\pm 0.07}$

Table 2: Test error rates for ResNets trained on ImageNet, with and without SAM.

Source: “SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION”

Experiment



Dataset

- “Obesity Levels”

- Data came from a study called:
  “Estimation of Obesity Levels with a Trained Neural Network Approach optimized by the Bayesian Technique”

- Made publicly available on Kaggle by an author of the study:
  Fatma Hilal Yagin

Labels

- Insufficient Weight, Normal Weight, Overweight Level I & II, Obesity Type I & II & III

- Obesity is measured by BMI = Weight /Height

- 7 different classes is more then necessary, but the focus of the experiment is on measuring the impact of different methods and hyperparameters on the model.

Features

- 16 different features
- 5 Biological
  (Age, Gender, Weight, Height, Family History of Obesity)
- 11 behavioral or socioeconomic
  Examples:
  Do you smoke/drink/count calories?
  Means of transportation or contact to digital devices.
- I set the feature weight to zero

Noise in Data

- “Inaccurate” Reporting in weight Loss Studies.

  “…group 1 underreported their actual food intake by an average (+/- SD) of 47 +/- 16 percent and overreported their physical activity by 51 +/- 75 percent.”) (1)

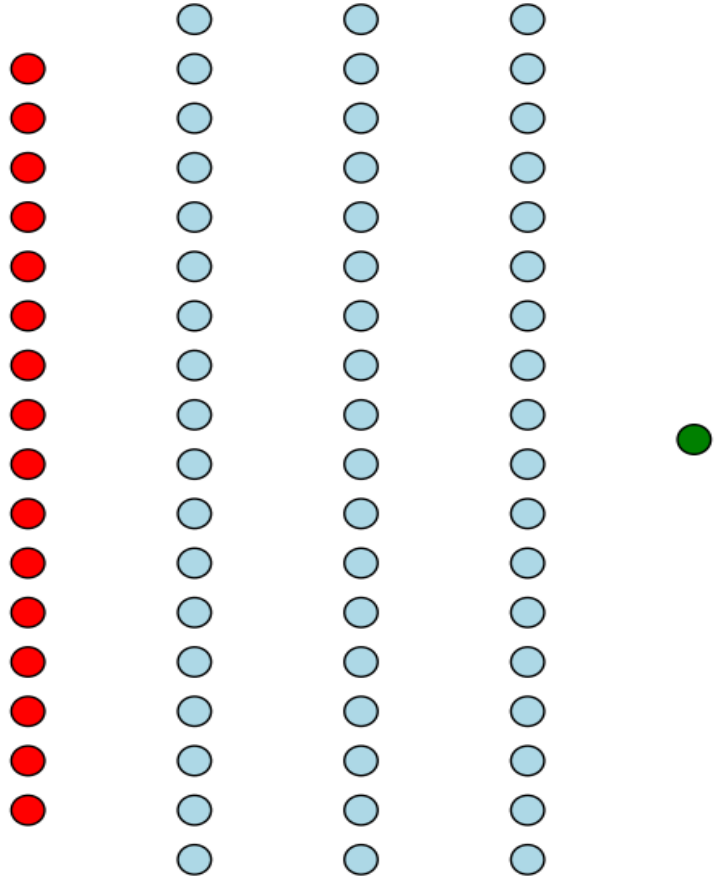
- Questionable methodology in studies.

(1)“Discrepancy between self-reported and actual caloric intake and exercise in obese subjects”

Model

4 Hidden Layers with 18 nodes each and ReLu activation function

16 Inputs



1 Output with Sigmoid activation function

Model

- Fixed Hyperparameters
  - Batch size: 32
  - Epochs: 200 (100 with SAM)
  - Cosine Annealing: T_max = 100
- Flexible Hyperparameters
  - Optimizer: SGD, Adam
  - lr : 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 2e-2
  - wd: 0, 1e-4, 1e-3, 1e-2
  - SAM : 0.02, 0.05, 0.1

Results

Procedure

- We train our model and then find the MSE of our model on the test data.

- We do this three times for any combination of flexible hyperparameters and average the MSE.

Apology

- I know that machine learning enthusiasts are deeply in love with graphs.
- But with 4 flexible hyperparameters this can’t be plotted.
- Tables will work better for this.

GPT4o : „Nerd in love with graphs“

SGD vs Adam (no SAM)

Adam no SAM	0 wd	1e-4 wd	1e-3 wd	1e-3 wd
3e-5 lr	2.12291149	1.89202972	2.22012893	1.93027723
1e-4 lr	1.49266938	1.43069557	1.46342095	1.58413537
3e-4 lr	1.21410422	1.06937583	1.25926745	1.15934475
1e-3 lr	1.01649876	0.90992079	0.9319943	0.97849667
3e-3 lr	0.946555	0.85182645	0.85965474	0.93860888
2e-2 lr	1.01257888	1.07348271	1.25872445	1.32351005

SGD no SAM	0 wd	1e-4 wd	1e-3 wd	1e-2 wd
3e-5 lr	3.62476587	3.61885889	3.6268123	3.53993098
1e-4 lr	3.08547036	2.8002034	2.73712643	3.04313159
3e-4 lr	1.74086797	1.99654186	1.89854797	1.95035009
1e-3 lr	1.4374007	1.30535924	1.35845566	1.54142503
3e-3 lr	1.29904699	1.73110859	1.34918125	1.88780828
2e-2 lr	2.62340593	2.72503956	2.44728176	2.39643057

SGD vs Adam (SAM 0.05)

AdamSAM.05	0 wd	1e-4 wd	1e-3 wd	1e-3 wd
3e-5 lr	3.49386414	3.5093224	3.53408011	3.57435648
1e-4 lr	2.73934921	2.42311056	2.75552972	2.71941233
3e-4 lr	1.90767531	1.92567043	1.83400416	2.0666304
1e-3 lr	1.36145969	1.31770722	1.18528076	1.35685186
3e-3 lr	1.02621742	1.03551219	1.06326592	1.27810097
2e-2 lr	2.19685857	2.87727813	2.86784319	1.50141843

SGD SAM.05	0 wd	1e-4 wd	1e-3 wd	1e-2 wd
3e-5 lr	3.63881938	3.69519416	3.68839312	3.70761967
1e-4 lr	3.64164702	3.59298952	3.65278379	3.6095411
3e-4 lr	3.35367568	3.51196758	3.40546719	3.42777014
1e-3 lr	2.85132233	2.99642221	2.96286003	2.94961119
3e-3 lr	2.28282213	2.48275042	2.32628258	2.23272216
2e-2 lr	2.25685358	2.14518758	2.5352362	2.76775114

SGD vs Adam (SAM 0.02)

AdamSAM.02	0 wd	1e-4 wd	1e-3 wd	1e-3 wd
3e-5 lr	3.10973732	3.40380311	3.29089387	3.22000496
1e-4 lr	2.30811572	2.24761065	2.22062127	2.06337472
3e-4 lr	1.80152082	1.5153414	1.61599731	1.71505082
1e-3 lr	1.09218804	1.05522124	1.10692116	1.05838398
3e-3 lr	0.93751236	0.9629524	0.94113191	0.97522306
2e-2 lr	1.27347386	1.11533934	1.21533879	1.36327453

SGDSAM.02	0 wd	1e-4 wd	1e-3 wd	1e-2 wd
3e-5 lr	3.69324319	3.6754214	3.65705713	3.68308051
1e-4 lr	3.55987899	3.54674697	3.47856069	3.53895752
3e-4 lr	3.18352477	3.15310915	3.16189623	3.20011465
1e-3 lr	2.07361627	2.0664585	2.49926194	2.60127719
3e-3 lr	1.58715832	2.49315588	1.92409738	1.93272591
2e-2 lr	2.09878623	2.90633154	2.59858378	2.28931363

SGD vs Adam (SAM 0.1)

AdamSAM0.1	0 wd	1e-4 wd	1e-3 wd	1e-3 wd
3e-5 lr	3.67636935	3.64313046	3.6133639	3.6835444
1e-4 lr	3.41977461	3.08524497	3.09248066	3.13601685
3e-4 lr	2.43300422	2.54177976	2.56951451	2.59420816
1e-3 lr	1.59123635	1.58435198	1.59936432	2.00447798
3e-3 lr	1.24811963	1.23226829	1.28906373	1.51639903
2e-2 lr	3.67359678	3.6839625	2.92754197	3.059793

SGDSAM 0.1	0 wd	1e-4 wd	1e-3 wd	1e-2 wd
3e-5 lr	3.68318947	3.71154213	3.68890675	3.66121133
1e-4 lr	3.66372172	3.66691033	3.65890185	3.63865503
3e-4 lr	3.59576265	3.65166354	3.62507184	3.63506643
1e-3 lr	3.55078491	3.47811484	3.50040158	3.55130442
3e-3 lr	3.10845327	3.01457723	2.9209319	2.88371253
2e-2 lr	2.34281007	2.52955349	2.70081941	2.80287854

Conclusion for our Model and Data

- Adam with a learning rate of 3e-3 works best and can be enhanced by weight decay (especially with wd 1e-4).
- The results don’t support the implementation of SAM in our Model.
- The results don’t support the implementation of SGD in our Model

In defense of SAM

- Adam+SAM(-WD) beats Adam (-WD).
  (0.93 to 0.94)

- We needed to halve its number of epochs to keep the time complexity and number of weight updates comparable.

- But too those of you who think I wronged SAM…

A Second Chance for SAM?

No.
Even with 200 epochs SAM did not set a new benchmark for Adam.
Not averaged and not even in one of the three tries!

AdamSAM.02	0 wd	1e-4 wd	1e-3 wd	1e-3 wd
3e-5 lr	2.77573427	2.68003885	2.81993278	2.70643028
1e-4 lr	1.94776758	1.79024291	1.86626244	1.83289369
3e-4 lr	1.39283121	1.21019701	1.19946941	1.40464489
1e-3 lr	0.95352995	0.90911883	0.93369329	1.00014228
3e-3 lr	0.88088795	0.96705212	0.91308735	0.94491996
2e-2 lr	1.39928099	1.07448526	1.39347351	1.46274924

Note of Thanks

- Big Thanks to my project advisor Antonio!
- Big Thanks to David Samuel whose Github Repo helped me to implement SAM!
- Big Thanks to my Steam Deck who trained 504 models for me!