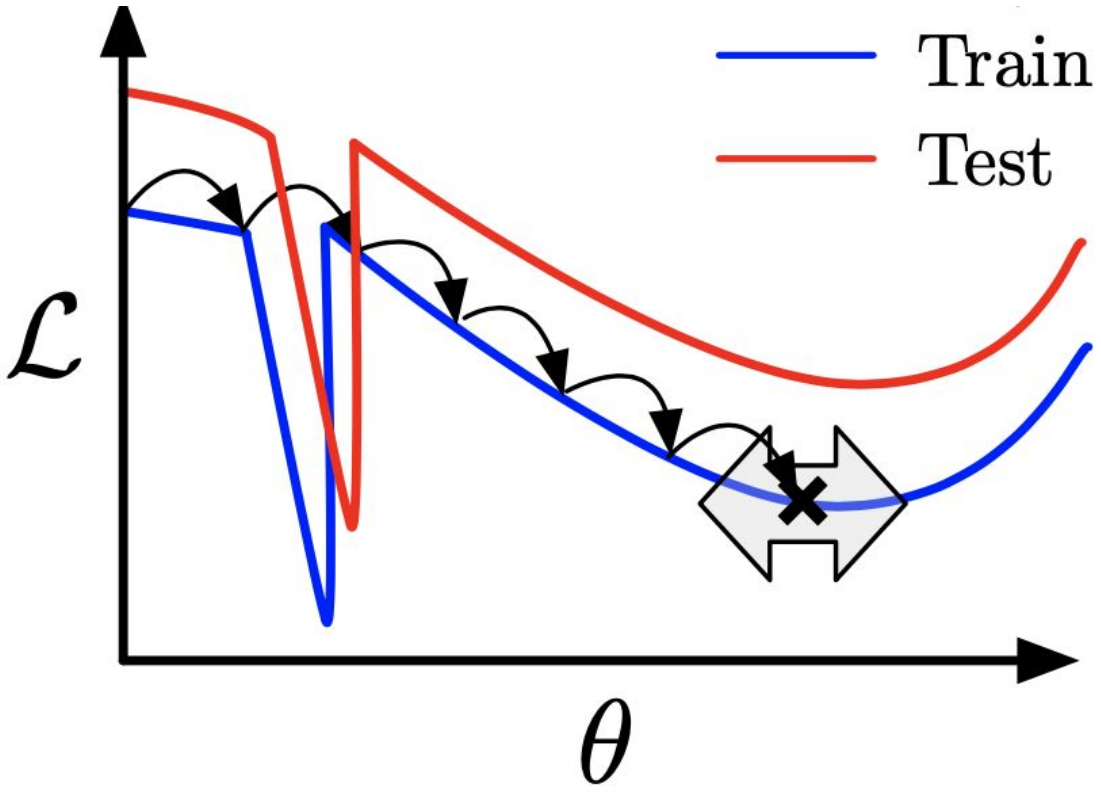
Широкие и узкие оптимумы у нейросетей

Писцов Георгий, БПМИ202

Что такое широкие и узкие оптимумы?



Идея: Сходимость к широким оптимумам ведёт к лучшей обобщенности(generalization) модели, так как при небольшом возмущении значение остается приблизительно оптимальным

В данной работе мы рассмотрим два алгоритма, которые позволяют нам искать широкие оптимумы: SWA и SAM

Наглядная визуализация узких и широких оптимумов

Анализ SGD

Мы рассматриваем циклический или константный шаг(вырожденный), заданный формулой:

$$\alpha(i) = (1 - t(i))\alpha_1 + t(i)\alpha_2,$$

$$t(i) = \frac{1}{c}\left(\mathrm{mod}(i-1, c) + 1\right).$$

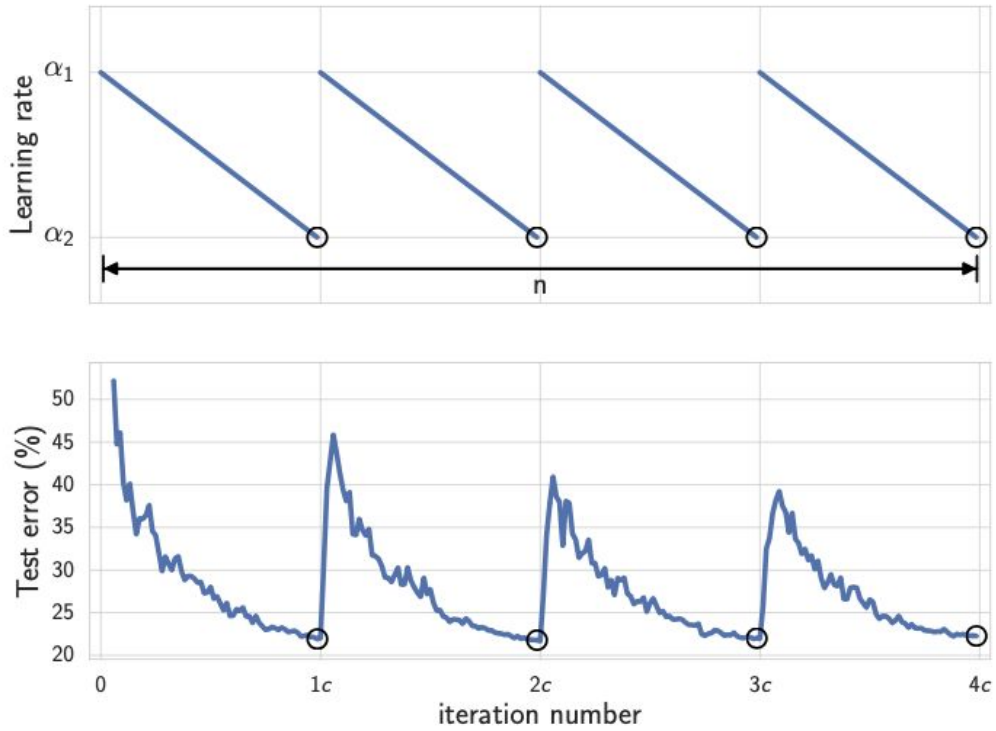


Figure 2: **Top**: cyclical learning rate as a function of iteration. **Bottom**: test error as a function of iteration for cyclical learning rate schedule with Preactivation-ResNet-164 on CIFAR-100. Circles indicate iterations corresponding to the minimum learning rates.

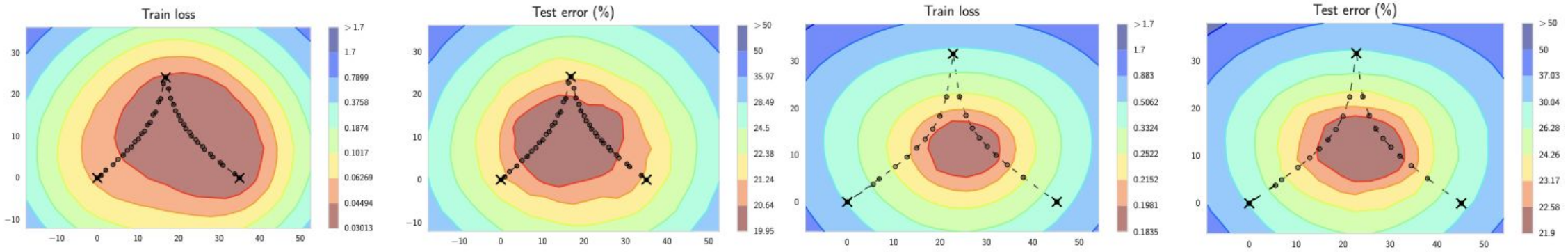


Figure 3: The L_2-regularized cross-entropy train loss and test error surfaces of a Preactivation ResNet-164 on CIFAR-100 in the plane containing the first, middle and last points (indicated by black crosses) in the trajectories with (**left two**) cyclical and (**right two**) constant learning rate schedules.

Замечаем, что:

1. Оба случая исследуют точки близко к периферии набора высокоэффективных сетей
2. Есть тот самый сдвиг между поверхностью потерь на тесте и на обучении
3. Усреднение наталкивает на мысль о возможном успехе

Алгоритм SWA

Algorithm 1 Stochastic Weight Averaging

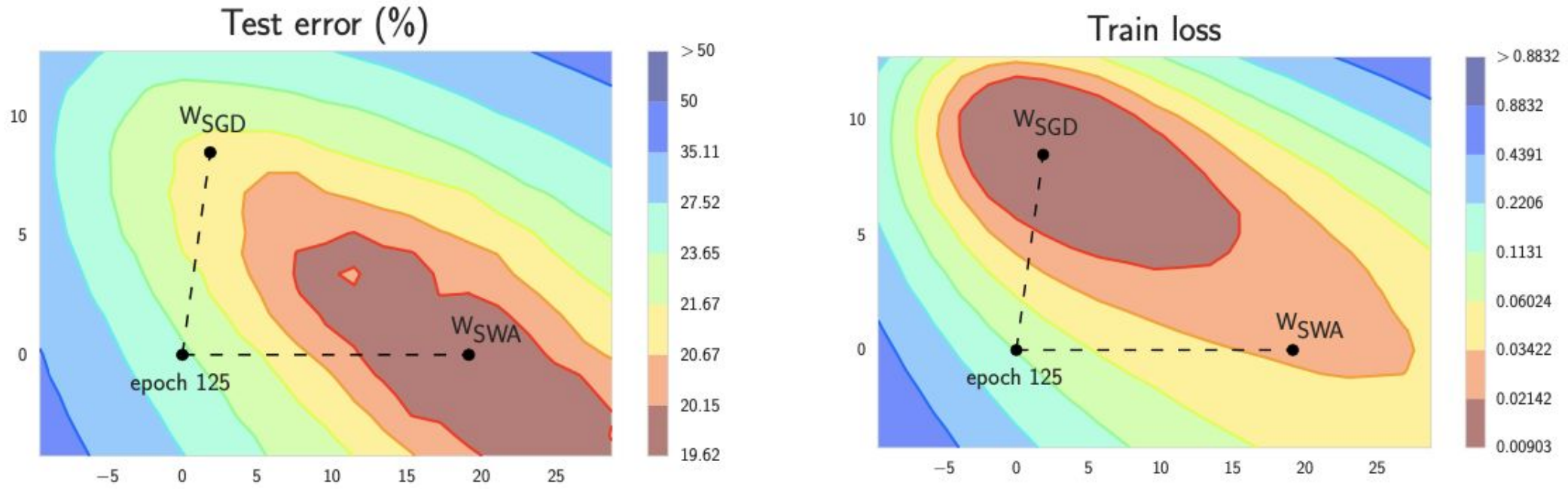
Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (for constant learning rate $c = 1$), number of iterations n

Ensure: w_{SWA}

$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\text{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}

Чуть позже мы увидим более современную версию алгоритма, которая вносит небольшую модификацию

Сравнение SWA и SGD



Иллюстрации SWA и SGD с преактивацией ResNet-164 на CIFAR-100. Слева и справа: поверхности ошибок теста и потерь обучения, показывающие веса, предложенные SGD (при сходимости) и SWA, начиная с той же инициализации SGD после 125 эпох обучения.

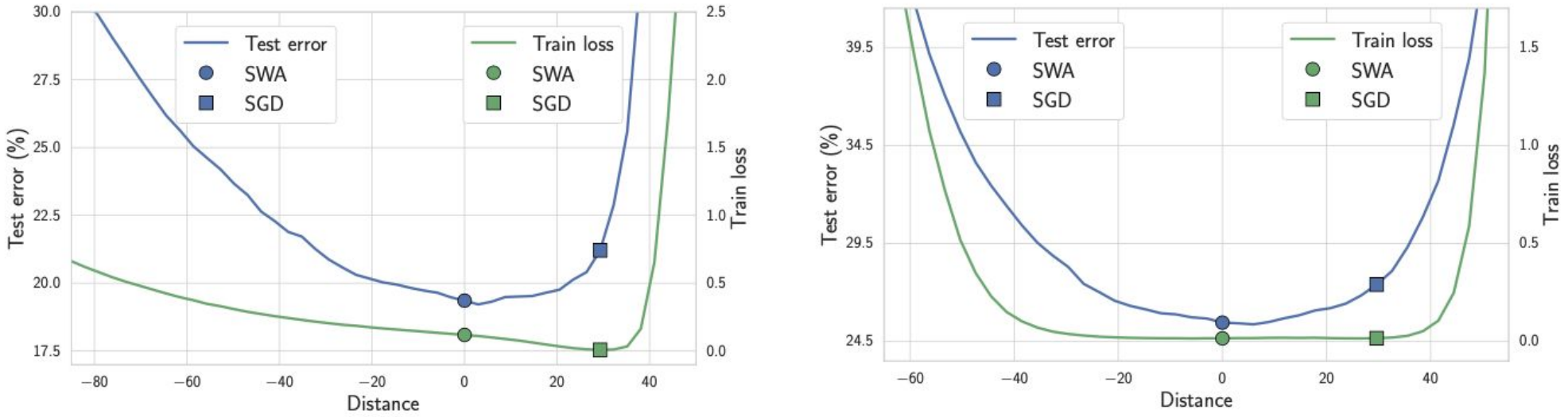


Figure 5: L_2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100. **Left**: Preactivation ResNet-164. **Right**: VGG-16.

Рассматриваем линейную интерполяцию::

$$w(t) = t \cdot w_{\text{SGD}} + (1 - t) \cdot w_{\text{SWA}}$$

Замечаем, что:

1. Графики ошибок действительно смещены
2. SWA находит более широкий оптимум
3. SGD находит оптимум, в одном из направлений которого ошибка резко растёт

Связь SWA и FGE: теория

- В первом случае усреднение весов
- Во втором случае усреднение прогнозов

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(w_i).$$

$$\Delta_i = w_i - w_{\mathrm{SWA}}$$

$$\Delta = \max_{i=1}^{n} \|\Delta_i\|$$

$$f(w_j) = f(w_{\mathrm{SWA}}) + \langle \nabla f(w_{\mathrm{SWA}}), \Delta_j \rangle + O(\|\Delta_j\|^2),$$

$$\bar{f} - f(w_{\mathrm{SWA}}) = \frac{1}{n}\sum_{i=1}^{n} \left(\langle \nabla f(w_{\mathrm{SWA}}), \Delta_i \rangle + O(\|\Delta_i\|^2) \right)$$

$$= \left\langle \nabla f(w_{\mathrm{SWA}}), \frac{1}{n}\sum_{i=1}^{n} \Delta_i \right\rangle + O(\Delta^2) = O(\Delta^2),$$

Связь SWA и FGE: практика

Мы запускаем FGE на 20 эпох и сравнили предсказания различных моделей на тестовом наборе данных с преактивацией ResNet-164 на CIFAR-100.
Затем мы усредняем веса предложений и вычисляем вероятности классов на тестовом наборе данных.
Для FGE и SWA доля идентично помеченных объектов составляет 95,26%.

Сравнение SGD, FGE и SWA

Table 1: Accuracies (%) of SWA, SGD and FGE methods on CIFAR-100 and CIFAR-10 datasets for different training budgets. Accuracies for the FGE ensemble are from Garipov et al. [2018].

			SWA		
DNN (Budget)	SGD	FGE (1 Budget)	1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-164 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 0.27
PyramidNet-272 (300)	83.41 ± 0.21	–	–	83.93 ± 0.18	84.16 ± 0.15
		CIFAR-10			
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18
ResNet-164 (150)	95.28 ± 0.10	95.45	95.56 ± 0.11	95.77 ± 0.04	95.83 ± 0.03
WRN-28-10 (200)	96.18 ± 0.11	96.36	96.45 ± 0.11	96.64 ± 0.08	96.79 ± 0.05
ShakeShake-2x64d (1800)	96.93 ± 0.10	–	–	97.16 ± 0.10	97.12 ± 0.06

Расписание шага SWA

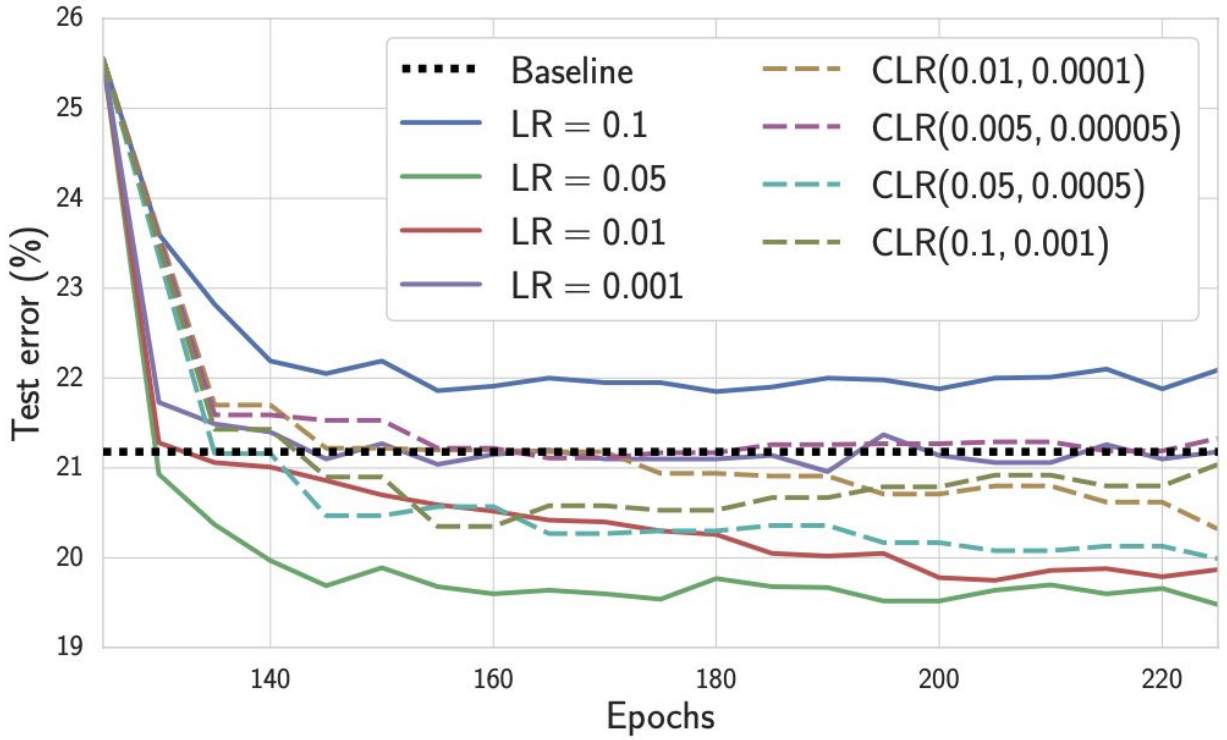


Figure 6: Test error as a function of training epoch for SWA with different learning rate schedules with a Preactivation ResNet-164 on CIFAR-100.

Другие подходы: SAM

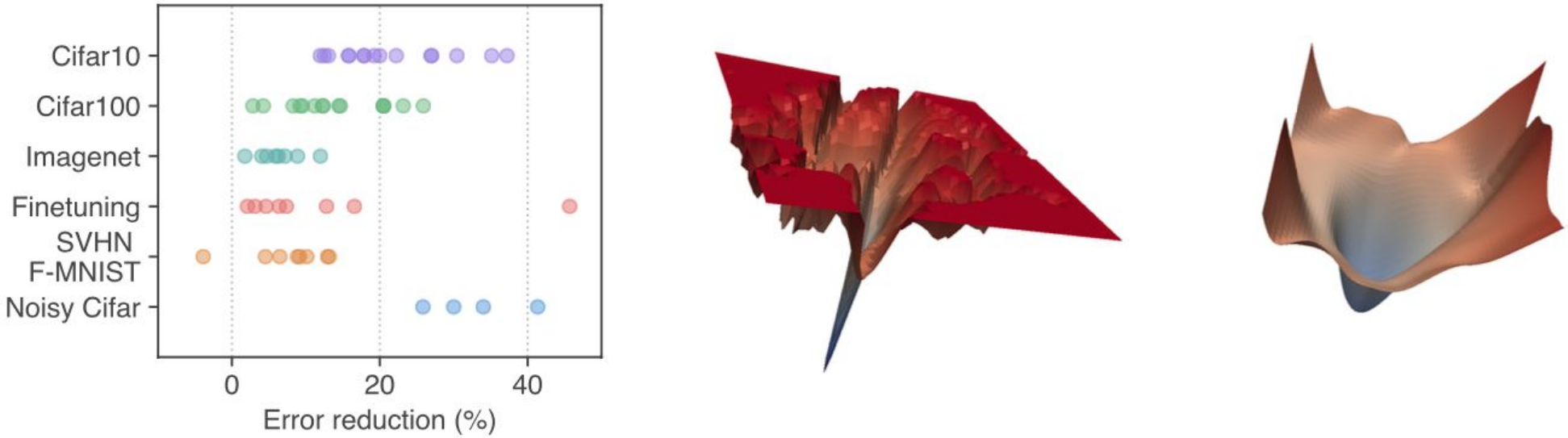


Figure 1: (left) Error rate reduction obtained by switching to SAM. Each point is a different dataset / model / data augmentation. (middle) A sharp minimum to which a ResNet trained with SGD converged. (right) A wide minimum to which the same ResNet trained with SAM converged.

Описание алгоритма SAM

Идея состоит в том, что вместо того, чтобы искать значения параметров w, которые просто имеют низкое значение потерь при обучении, мы ищем значения параметров, все окрестности которых имеют равномерно низкое значение потерь при обучении. Для этого мы будем решать минимаксную задачу

$$\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}),$$

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \, \text{sign}\left(\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right) |\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|^{q-1} / \left(\|\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\|_q^q \right)^{1/p}$$

$$\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \approx \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(w)|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}.$$

SAM реализация

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, 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

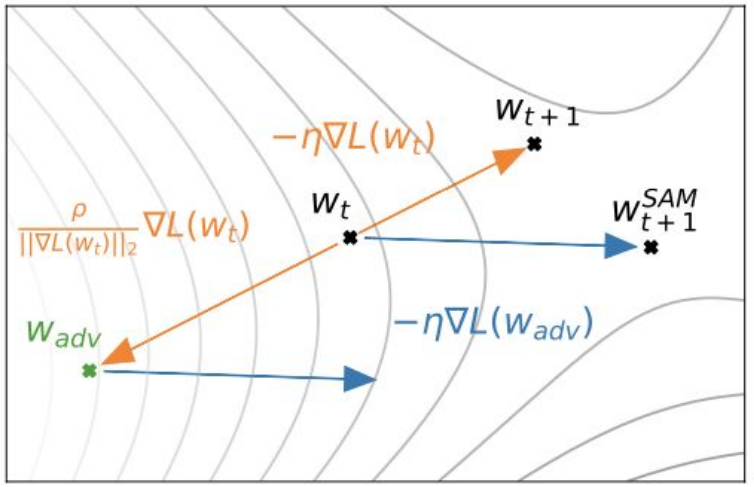


Figure 2: Schematic of the SAM parameter update.

Сравнение SAM с SGD на примерах

		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).

Устойчивость к шуму

Method	Noise rate (%)			
	20	40	60	80
Sanchez et al. (2019)	94.0	92.8	90.3	74.1
Zhang & Sabuncu (2018)	89.7	87.6	82.7	67.9
Lee et al. (2019)	87.1	81.8	75.4	-
Chen et al. (2019)	89.7	-	-	52.3
Huang et al. (2019)	92.6	90.3	43.4	-
MentorNet (2017)	92.0	91.2	74.2	60.0
Mixup (2017)	94.0	91.5	86.8	76.9
MentorMix (2019)	**95.6**	**94.2**	91.3	**81.0**
SGD	84.8	68.8	48.2	26.2
Mixup	93.0	90.0	83.8	70.2
Bootstrap + Mixup	93.3	92.0	87.6	72.0
SAM	95.1	93.4	90.5	77.9
Bootstrap + SAM	95.4	**94.2**	**91.8**	79.9

Table 4: Test accuracy on the clean test set for models trained on CIFAR-10 with noisy labels. Lower block is our implementation, upper block gives scores from the literature, per Jiang et al. (2019).

Тот факт, что SAM ищет параметры модели, устойчивые к возмущениям, позволяет предположить, что SAM может быть устойчив к шуму в обучающем множестве (который может возмутить ландшафт потерь при обучении).

Новые подходы

Algorithm 1 Stochastic Weight Averaging [48]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, averaging start epoch E, averaging frequency ν, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.

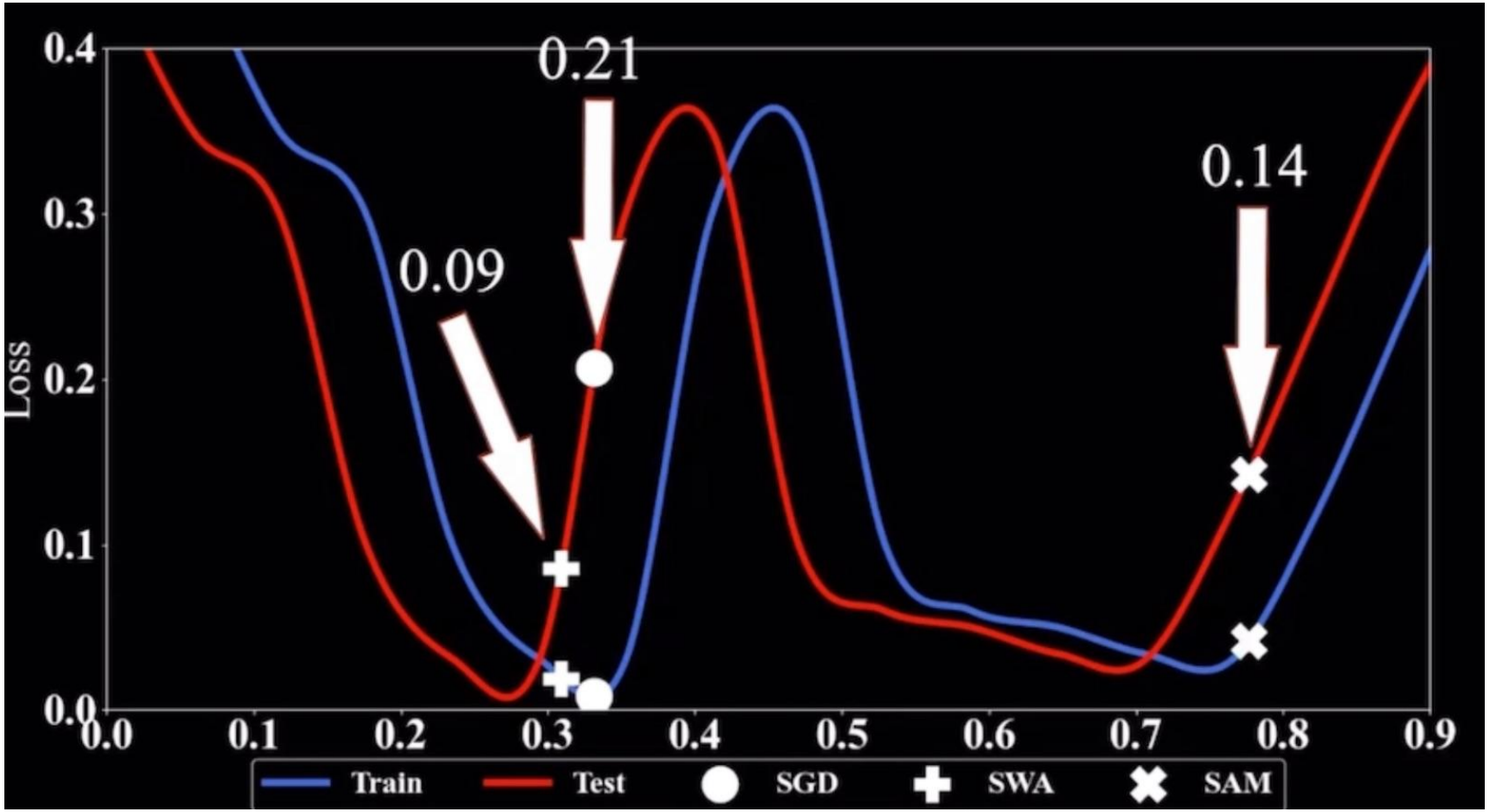
for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t\right)$
  Update parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\boldsymbol{g}$
  if $k \geq E$ and $\mathrm{mod}(k, \nu) = 0$ **then**
    $\boldsymbol{\theta}_{t+1}^{\mathrm{SWA}} = \left(\boldsymbol{\theta}_t^{\mathrm{SWA}} \cdot l + \boldsymbol{\theta}_{t+1}^{\mathrm{SWA}}\right) / (l+1)$
  end if
end for
return $\boldsymbol{\theta}^{\mathrm{SWA}}$

Algorithm 2 Sharpness-Aware Minimization [22]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, neighborhood radius ρ, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.

for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute worst-case perturbation $\widehat{\boldsymbol{\epsilon}} \leftarrow \rho \frac{\nabla\mathcal{L}(\boldsymbol{\theta})}{\|\nabla\mathcal{L}(\boldsymbol{\theta})\|_2}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t^{\mathrm{SAM}} + \widehat{\boldsymbol{\epsilon}}\right)$
  Update parameters $\boldsymbol{\theta}_{t+1}^{\mathrm{SAM}} \leftarrow \boldsymbol{\theta}_t^{\mathrm{SAM}} - \eta\boldsymbol{g}$
end for
return $\boldsymbol{\theta}^{\mathrm{SAM}}$

Чем же отличаются оптимумы SAM и SWA



Чем же отличаются оптимумы SAM и SWA

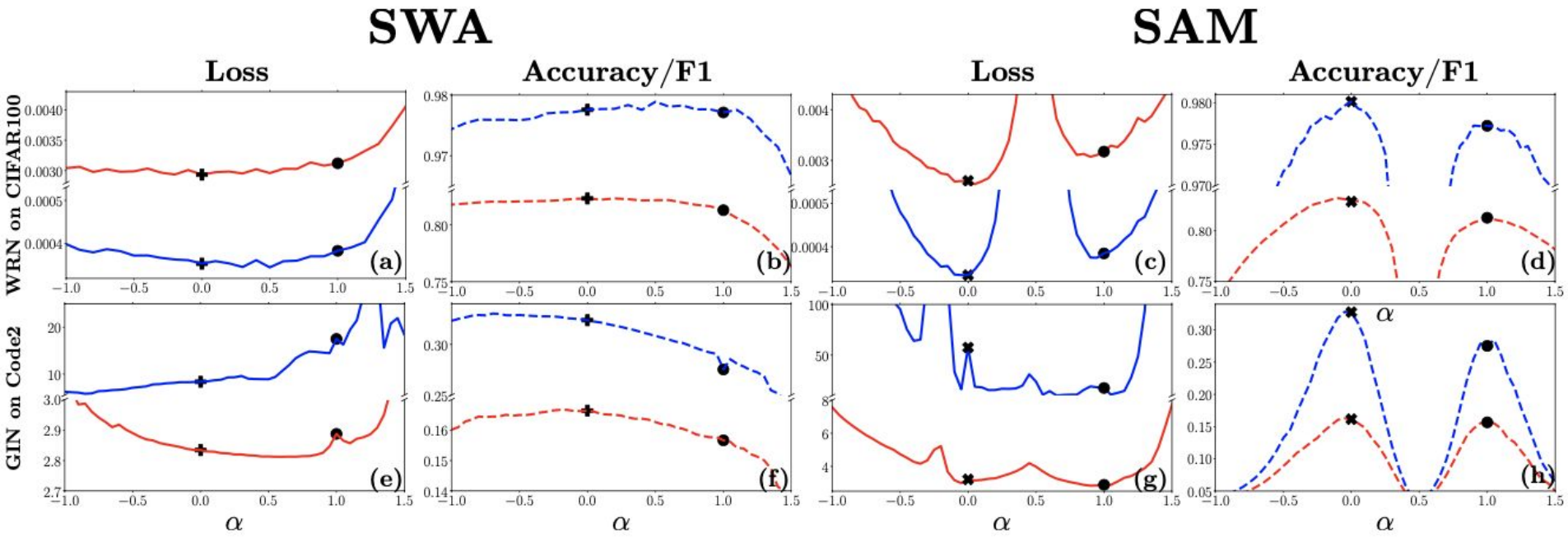


Figure 2: Training (blue) and test (red) losses (—) and accuracies (·····) of linear interpolations $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta} + \alpha\boldsymbol{\theta}'$ (for $\alpha \in [-1, 1.5]$) between SWA (+) and SAM (×) solutions ($\alpha = 0.0$) and non-flat baseline solutions (●, $\alpha = 1.0$).

Сравниваем все на примерах

Table 2: CV test results: Supervised Classification (SC), and Self-Supervised Learning (SSL) tasks.

Task	Model	Baseline	SWA	SAM	SWA+SAM
SC: CIFAR10	WRN-28-10	$96.78_{\pm 0.03}$	$-0.05_{\pm 0.04}$	$\mathbf{+0.34}_{\pm 0.09}$	$+0.25_{\pm 0.05}$
	PN-272	$96.73_{\pm 0.14}$	$+0.22_{\pm 0.14}$	$\mathbf{+0.42}_{\pm 0.06}$	$\mathbf{+0.41}_{\pm 0.02}$
	ViT-B-16	$98.95_{\pm 0.02}$	$-0.04_{\pm 0.04}$	$+0.07_{\pm 0.01}$	$\mathbf{+0.10}_{\pm 0.01}$
	Mixer-B-16	$96.65_{\pm 0.03}$	$+0.02_{\pm 0.03}$	$\mathbf{+0.19}_{\pm 0.05}$	$\mathbf{+0.22}_{\pm 0.06}$
SC: CIFAR100	WRN-28-10	$80.93_{\pm 0.19}$	$+1.62_{\pm 0.06}$	$+1.82_{\pm 0.14}$	$\mathbf{+2.24}_{\pm 0.14}$
	PN-272	$80.86_{\pm 0.12}$	$+1.88_{\pm 0.04}$	$+2.33_{\pm 0.08}$	$\mathbf{+2.60}_{\pm 0.09}$
	ViT-B-16	$92.77_{\pm 0.07}$	$-0.12_{\pm 0.05}$	$\mathbf{+0.19}_{\pm 0.09}$	$\mathbf{+0.13}_{\pm 0.07}$
	Mixer-B-16	$83.77_{\pm 0.08}$	$+0.45_{\pm 0.06}$	$+0.52_{\pm 0.15}$	$\mathbf{+0.97}_{\pm 0.12}$
SSL: CIFAR10	MoCo	$\mathbf{89.25}_{\pm \mathbf{0.07}}$	$-0.03_{\pm 0.10}$	$-0.25_{\pm 0.06}$	$-0.17_{\pm 0.10}$
	SimCLR	$\mathbf{88.66}_{\pm 0.08}$	$-0.05_{\pm 0.06}$	$\mathbf{+0.05}_{\pm 0.04}$	$-0.13_{\pm 0.06}$
	SimSiam	$\mathbf{89.86}_{\pm \mathbf{0.22}}$	$\mathbf{+0.12}_{\pm \mathbf{0.26}}$	$\mathbf{+0.07}_{\pm 0.10}$	$\mathbf{+0.11}_{\pm 0.10}$
	BarlowTwins	$\mathbf{86.34}_{\pm \mathbf{0.24}}$	$-0.09_{\pm 0.19}$	$\mathbf{+0.09}_{\pm 0.15}$	$\mathbf{+0.14}_{\pm 0.05}$
	BYOL	$90.32_{\pm 0.14}$	$\mathbf{+0.70}_{\pm 0.05}$	$+0.14_{\pm 0.03}$	$+0.21_{\pm 0.07}$
	SwaV	$87.28_{\pm 0.05}$	$\mathbf{+0.09}_{\pm \mathbf{0.06}}$	$\mathbf{+0.07}_{\pm \mathbf{0.12}}$	$+0.02_{\pm 0.06}$
SSL: ImageNette	MoCo	$81.74_{\pm 0.18}$	$+0.97_{\pm 0.10}$	$+0.91_{\pm 0.32}$	$\mathbf{+1.40}_{\pm 0.10}$
	SimCLR	$83.28_{\pm 0.22}$	$\mathbf{+0.95}_{\pm \mathbf{0.25}}$	$+0.18_{\pm 0.24}$	$\mathbf{+1.07}_{\pm \mathbf{0.13}}$
	SimSiam	$81.77_{\pm 0.14}$	$\mathbf{+0.20}_{\pm \mathbf{0.37}}$	$\mathbf{+0.33}_{\pm \mathbf{0.28}}$	$\mathbf{+0.18}_{\pm \mathbf{0.26}}$
	BarlowTwins	$77.49_{\pm 0.36}$	$+0.20_{\pm 0.16}$	$\mathbf{+0.47}_{\pm \mathbf{0.27}}$	$\mathbf{+0.66}_{\pm \mathbf{0.57}}$
	BYOL	$84.16_{\pm 0.14}$	$\mathbf{+0.76}_{\pm \mathbf{0.08}}$	$+0.15_{\pm 0.25}$	$+0.31_{\pm 0.19}$
	SwaV	$88.16_{\pm 0.31}$	$\mathbf{+1.04}_{\pm \mathbf{0.27}}$	$+0.03_{\pm 0.10}$	$\mathbf{+1.03}_{\pm \mathbf{0.09}}$

Сравниваем все на примерах

Figure 4: (a) NLP test results: Open-Domain Question Answering and Natural Language Understanding (GLUE) including paraphrase, sentiment analysis, and textual entailment. (b) GRL test results: Node Property Prediction (NPP), Graph Property Prediction (GPP), Link Property Prediction (LPP).

Task	Model	Baseline	SWA	SAM	SWA+SAM
NQ	FiD	$49.35_{\pm 0.44}$	$-0.20_{\pm 0.33}$	$\mathbf{+0.33_{\pm 0.19}}$	$\mathbf{+0.48_{\pm 0.21}}$
TriviaQA	FiD	$67.74_{\pm 0.29}$	$+0.40_{\pm 0.24}$	$\mathbf{+0.89_{\pm 0.03}}$	$\mathbf{+0.92_{\pm 0.10}}$
COLA	RoBERTa	$60.41_{\pm 0.22}$	$+0.09_{\pm 0.08}$	$\mathbf{+1.57_{\pm 1.20}}$	$\mathbf{+1.41_{\pm 1.14}}$
SST	RoBERTa	$94.95_{\pm 0.13}$	$-0.30_{\pm 0.27}$	$-0.23_{\pm 0.40}$	$\mathbf{+0.19_{\pm 0.14}}$
MRPC	RoBERTa	$89.14_{\pm 0.57}$	$+0.08_{\pm 0.49}$	$\mathbf{+0.73_{\pm 0.43}}$	$\mathbf{+0.81_{\pm 0.38}}$
STSB	RoBERTa	$90.40_{\pm 0.02}$	$+0.00_{\pm 0.05}$	$\mathbf{+0.38_{\pm 0.17}}$	$\mathbf{+0.35_{\pm 0.16}}$
QQP	RoBERTa	$91.36_{\pm 0.07}$	$+0.01_{\pm 0.06}$	$\mathbf{+0.08_{\pm 0.07}}$	$\mathbf{+0.06_{\pm 0.08}}$
MNLI	RoBERTa	$87.41_{\pm 0.09}$	$+0.08_{\pm 0.11}$	$\mathbf{+0.39_{\pm 0.02}}$	$+0.35_{\pm 0.03}$
QNLI	RoBERTa	$92.96_{\pm 0.06}$	$-0.08_{\pm 0.11}$	$+0.09_{\pm 0.01}$	$\mathbf{+0.11_{\pm 0.06}}$
RTE	RoBERTa	$80.09_{\pm 0.23}$	$-0.23_{\pm 0.20}$	$\mathbf{+0.70_{\pm 0.65}}$	$-0.46_{\pm 0.12}$

(a)

Task	Model	Baseline	SWA	SAM	SWA+SAM
NPP: Proteins	SAGE	$\mathbf{77.79_{\pm 0.18}}$	$\mathbf{-0.17_{\pm 0.22}}$	$\mathbf{-0.02_{\pm 0.13}}$	$\mathbf{-0.11_{\pm 0.15}}$
	DGCN	$\mathbf{85.42_{\pm 0.17}}$	$\mathbf{+0.11_{\pm 0.08}}$	$-0.14_{\pm 0.05}$	$-0.08_{\pm 0.07}$
NPP: Products	SAGE	$78.92_{\pm 0.08}$	$+0.39_{\pm 0.10}$	$+0.13_{\pm 0.08}$	$\mathbf{+0.57_{\pm 0.03}}$
	DGCN	$73.88_{\pm 0.13}$	$\mathbf{+0.44_{\pm 0.14}}$	$+0.08_{\pm 0.09}$	$\mathbf{+0.53_{\pm 0.05}}$
GPP: Code2	GCN	$16.04_{\pm 0.09}$	$+0.73_{\pm 0.11}$	$+0.36_{\pm 0.08}$	$\mathbf{+0.93_{\pm 0.15}}$
	GIN	$15.73_{\pm 0.11}$	$+0.83_{\pm 0.11}$	$+0.57_{\pm 0.09}$	$\mathbf{+1.10_{\pm 0.09}}$
GPP: Molpcba	GIN	$28.10_{\pm 0.11}$	$\mathbf{+0.40_{\pm 0.18}}$	$-0.33_{\pm 0.14}$	$\mathbf{+0.33_{\pm 0.16}}$
	DGCN	$25.65_{\pm 0.13}$	$\mathbf{+1.90_{\pm 0.20}}$	$-0.13_{\pm 0.18}$	$+1.34_{\pm 0.12}$
LPP: Biokg	CP	$84.06_{\pm 0.00}$	$\mathbf{+0.07_{\pm 0.01}}$	$0.00_{\pm 0.03}$	$\mathbf{+0.08_{\pm 0.02}}$
	ComplEx	$84.94_{\pm 0.01}$	$\mathbf{+0.14_{\pm 0.01}}$	$-0.02_{\pm 0.01}$	$\mathbf{+0.12_{\pm 0.02}}$
LPP: Citation2	GCN	$79.52_{\pm 0.41}$	$-0.05_{\pm 0.52}$	$+1.32_{\pm 0.06}$	$\mathbf{+1.50_{\pm 0.13}}$
	SAGE	$81.95_{\pm 0.02}$	$\mathbf{+1.15_{\pm 0.02}}$	$-0.31_{\pm 0.07}$	$+0.86_{\pm 0.04}$

(b)

Основные выводы:

1. Оптимизаторы поиска широкого оптимума могут уступать по ошибке другим оптимизаторам
2. SWA+SAM более устойчив и в 39/42 случаях показал результат лучше чем хотя бы один из SWA или SAM по отдельности
3. Рассмотренные оптимизаторы предлагают асимметричную отдачу: в худшем случае они снижают производительность на -0,30%, в лучшем - повышают на 2,60%.

Источники

SAM: https://arxiv.org/abs/2010.01412

SWA: https://arxiv.org/pdf/1803.05407.pdf

SAM+SWA вместе: https://arxiv.org/abs/2202.00661