Training DNNs in Low-Dimensional Subspaces

Master Colloquium | Janik Philipps

RWTH AACHEN UNIVERSITY

Motivation 1/2: Deep neural networks depend on million of parameters causing severe problems.

model	#params
SqueezeNet [18]	0.78M
ShuffleNetv2 [19]	1.36M
MobileNet [20]	3.32M
EfficientNet [21]	4.14M
GoogLeNet [22]	6.40M
DenseNet121 [23]	7.05M
SEResNet18 [24]	11.31M
Xception [25]	21.01M
Inceptionv3 [26]	22.32M
VGG11_bn [27]	28.52M

- Potential overfitting of the data
- Computationally and time intensive training
- Many training data needed
- Questionable solution quality

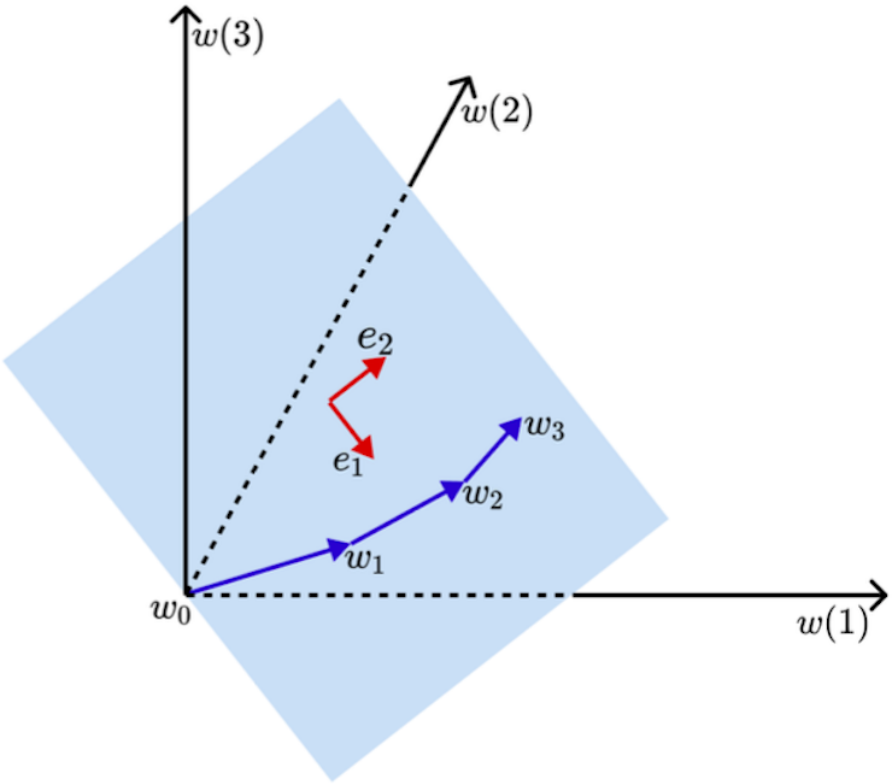
Motivation 2/2: The number of independent optimization variables may be smaller than the number of parameters.

- Due to strong mutual relationships, **regarding each parameter** of deep neural networks **as an independent variable is too rough**.
- The **gradients of parameters are strongly related** due to the training via backpropagation.
- The parameters in the same layers also have **synergy correlations**.

$$\Downarrow$$

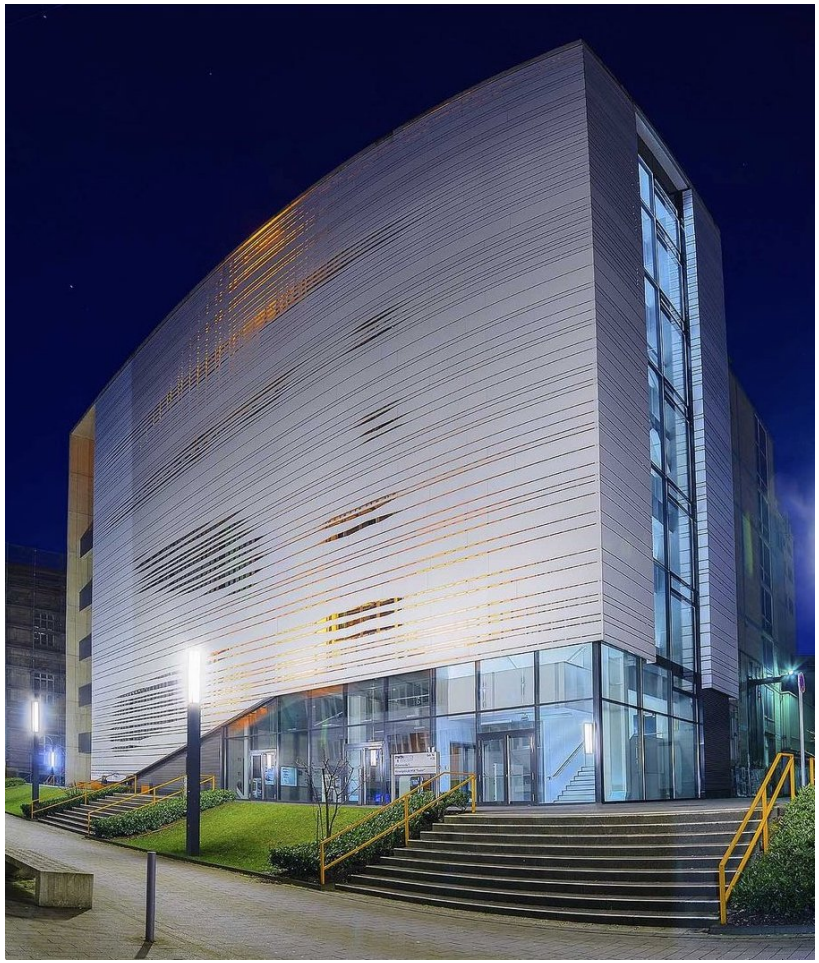
The Low-Dimensonal Trajectory Hypothesis:
For a neural network with n parameters, the parameters' trajectory over training can be approximately covered by a d-dimensional space with $d \ll n$. [1]

Illustration of the low-dimensional trajectory hypothesis



- There are **three parameters** $w(1)$, $w(2)$, $w(3)$ **to optimize**.
- The training trajectory $\{w_i\}_{i=0,\ldots,t}$ could be in a **two-dimensional subspace** spanned by e_1 and e_2.
- Training in the low-dimensional space can have **comparable performance** as training in the regular space.

Agenda



Theory

- Notation
- The Neural Tangent Kernel
- Theoretical Motivation

Practice

- Dynamic Linear Dimensionality Reduction (DLDR)
- Projected SGD (P-SGD)
- Numerical Experiments

Notation 1/2

- We investigate a single output neural network

$$f : \mathbb{R}^{n_0} \times \mathbb{R}^n \to \mathbb{R} : (x, w) \mapsto f(x, w).$$

- In order to train the network we require a training set

$$\mathcal{X} = \{(x_i, y_i) \mid i = 1, \ldots, m\} \subset \mathbb{R}^{n_0} \times \mathbb{R}.$$

- Also, we require a differentiable loss function

$$\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+ : (x, y) \mapsto \ell(x, y).$$

- Training the network is accomplished by minimizing the empirical error

$$E : \mathbb{R}^n \to \mathbb{R}_+ : w \mapsto \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big).$$

Notation 2/2

- The training dynamics are governed by the differential equation

$$\partial_t w_t = -\nabla E(w_t) = -\sum_{i=1}^{m} \nabla_w f(x_i, w_t) \cdot \nabla_x \ell\big(f(x_i, w_t), y_i\big).$$

- Written in terms of vectors and matrices this is equivalent to

$$\partial_t w_t = -\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E.$$

- Here we used the notations

$$\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} := \Big[\nabla_w f(x_1, w_t), \ldots, \nabla_w f(x_m, w_t)\Big] \in \mathbb{R}^{n \times m},$$

$$\nabla_{f(\mathcal{X}, w_t)} E := \Big[\nabla_x \ell\big(f(x_1, w_t), y_1\big), \ldots, \nabla_x \ell\big(f(x_m, w_t), y_m\big)\Big]^\mathsf{T} \in \mathbb{R}^m.$$

The Neural Tangent Kernel (NTK)

- The Neural Tangent Kernel refers to the matrix

$$\Theta_w = \nabla_w f(\mathcal{X}, w) \cdot \nabla_w f(\mathcal{X}, w)^\mathsf{T}.$$

- According to [2], in the **infinite-width limit**, it holds that

$$\nabla_w f(\mathcal{X}, w_t) \cdot \nabla_w f(\mathcal{X}, w_t)^\mathsf{T} = \Theta_{w_t} = \Theta_{w_0} = \nabla_w f(\mathcal{X}, w_0) \cdot \nabla_w f(\mathcal{X}, w_0)^\mathsf{T}.$$

- Using a singular value decomposition $\nabla_w f(\mathcal{X}, w_0) = U_0 \Sigma_0 V_0^\mathsf{T}$, yields

$$\Theta_{w_0} = \nabla_w f(\mathcal{X}, w_0) \cdot \nabla_w f(\mathcal{X}, w_0)^\mathsf{T} = U_0 \Sigma_0 \Sigma_0^\mathsf{T} U_0^\mathsf{T}.$$

- According to [3] and [4], the limiting NTK is low-rank, so Σ_0 is low-rank as well.
- Thus there exist $d \in \mathbb{N}$, $\tilde{U}_0 := [e_1, \ldots, e_d] \in \mathbb{R}^{m \times d}$, $\tilde{V}_0 := [e_1, \ldots, e_d] \in \mathbb{R}^{n \times d}$ and $\tilde{\Sigma}_0 = \operatorname{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d \times d}$ with

$$\Sigma_0 \approx \tilde{U}_0 \tilde{\Sigma}_0 \tilde{V}_0^\mathsf{T}.$$

Theoretical Motivation

Assumption: The layer width is unlimited.

- Using the properties of the NTK, we derive

$$\partial_t w_t = -\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E = -\nabla_w f(\mathcal{X}, w_0)^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E.$$

- Also, since the NTK is low-rank, we have

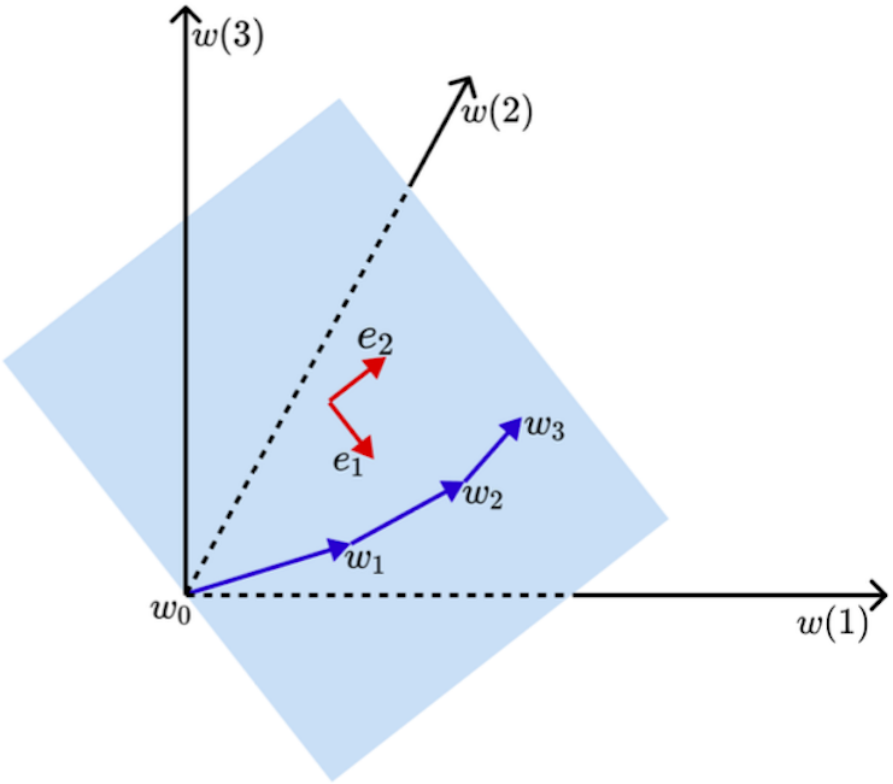
$$\nabla_w f(\mathcal{X}, w_0) = U_0 \Sigma_0 V_0^\mathsf{T} \approx U_0 \tilde{U}_0 \tilde{\Sigma}_0 \tilde{V}_0^\mathsf{T} V_0^\mathsf{T}.$$

- As a consequence, the dynamics of gradient flow are approximately governed by

$$\partial_t w_t = -\nabla_w f(\mathcal{X}, w_0)^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E \approx -V_0 \tilde{V}_0 \left(\tilde{\Sigma}_0 \tilde{U}_0^\mathsf{T} U_0^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E \right).$$

$\Rightarrow$ The training dynamics $\partial_t w_t$ of dimension n could be well embedded just in a d-dimensional space.

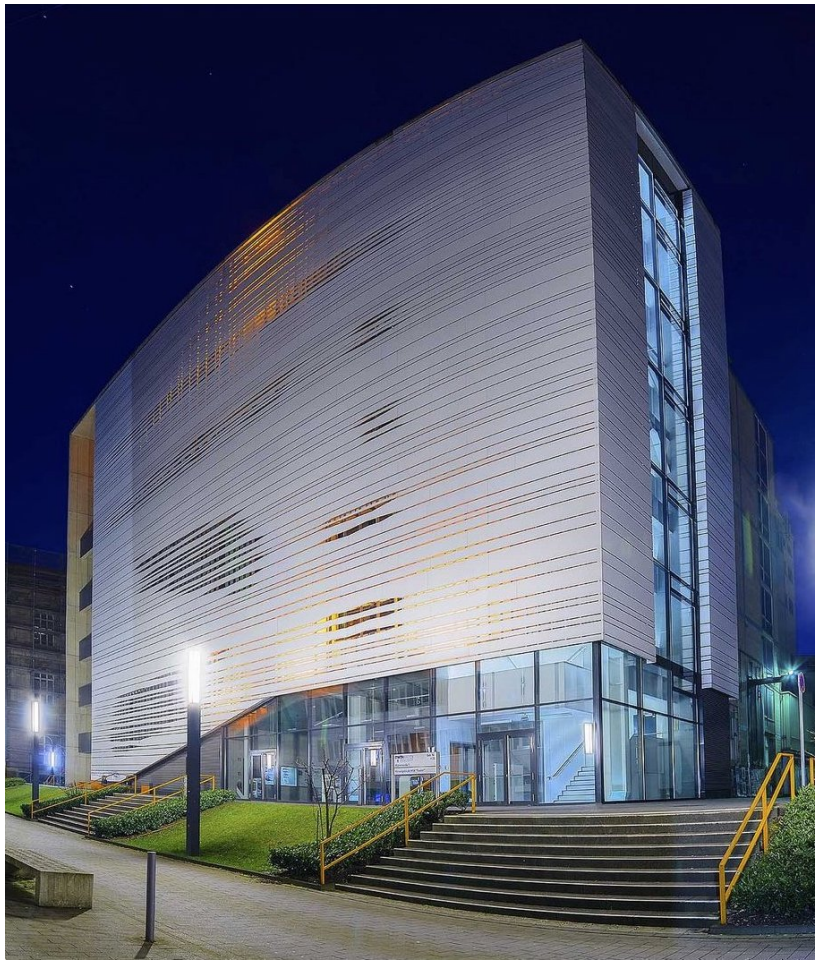
Illustration of the low-dimensional trajectory hypothesis



- $V_0 \tilde{V}_0$ refers to the the orthonormal basis of the subspace
- $\tilde{\Sigma}_0 \tilde{U}_0^\mathsf{T} U_0^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E$ describes the projected gradient

$$\partial_t w_t = -\nabla E(w_t) = -\nabla_w f(\mathcal{X}, w_0)^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E \approx -V_0 \tilde{V}_0 \left(\tilde{\Sigma}_0 \tilde{U}_0^\mathsf{T} U_0^\mathsf{T} \cdot \nabla_{f(\mathcal{X}, w_t)} E \right)$$

Agenda



Theory

- Notation
- The Neural Tangent Kernel
- Theoretical Motivation

Practice

- Dynamic Linear Dimensionality Reduction (DLDR)
- Projected SGD (P-SGD)
- Numerical Experiments

Low-Dimensional Subspace Extraction

Algorithm 7 Dynamic Linear Dimensionality Reduction (DLDR)

Input: Dimensionality of the subspace d, number of samples $t \geqslant d$

1: Sample parameter trajectory $(w_k)_{k=0}^{t}$ along the optimization trajectory;

2: Compute the mean $\overline{w} \coloneqq \frac{1}{t+1} \sum_{k=0}^{t} w_k$;

3: Centralize the samples as $W = [w_0 - \overline{w}, \ldots, w_t - \overline{w}]$;

4: Perform spectral decomposition such that $W^\mathsf{T} W = V \Sigma^2 V^{-1}$;

5: Obtain the d largest eigenvalues $\left[\sigma_1^2, \ldots, \sigma_d^2\right]$ with eigenvectors $\left[v_1, \ldots, v_d\right]$;

6: Determine the singular vectors $u_i = \frac{1}{\sigma_i} W v_i$ for $i = 1, \ldots, d$;

7: Return the orthonormal basis $\left[u_1, \ldots, u_d\right]$ of the subspace;

Training in Low-Dimensional Subspaces

Algorithm 8 Projected Stochastic Gradient Descent (P-SGD)

Input: Differentiable $E : \mathbb{R}^n \to \mathbb{R}$, batch size $b \in \mathbb{N}$, number of epochs $N \in \mathbb{N}$, $w_0 \in \mathbb{R}^n$, orthonormal basis $Q \in \mathbb{R}^{n \times d}$ of the d-dimensional subspace

1: Project $w_0 \leftarrow QQ^\mathsf{T} w_0$;

2: **for** $k = 0, \ldots, \frac{Nm}{b}$ **do**:

3: Draw a random subset $\mathcal{I}_k \subset \{1, \ldots, m\}$ of size $|\mathcal{I}_k| = b$;

4: Choose a step size $\eta_k > 0$;

5: Set $w_{k+1} \leftarrow w_k - \eta_k \cdot QQ^\mathsf{T} \nabla E(w_k, \mathcal{I}_k)$;

6: **end for**

Optimizing ResNet8 in a 15-dimensional space can achieve comparable performance as regular training in the full parameter space.

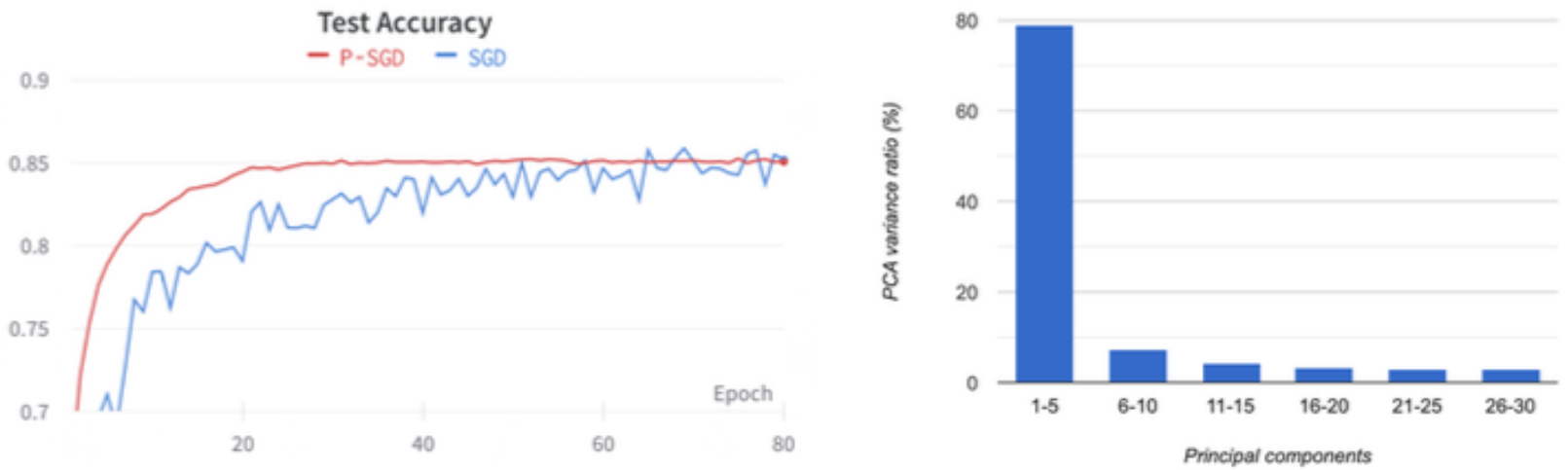


Figure 5: Experiments for ResNet8 on CIFAR-10. (a) Training performance of SGD and P-SGD in 15D subspace. (b) PCA variance ratio of the samples $W = [w_1, \ldots, w_{30}]$.

Optimizing DNNs in 40-dimensional spaces can achieve comparable performance as regular training in the full parameter space.

model	#params	SGD #training epochs			P-SGD #sampling epochs	
		50	100	200	50	100
SqueezeNet [18]	0.78M	60.96	62.49	71.14	69.45	71.70
ShuffleNetv2 [19]	1.36M	63.84	64.53	72.57	72.13	73.54
MobileNet [20]	3.32M	58.69	59.84	67.94	67.01	68.76
EfficientNet [21]	4.14M	62.53	63.44	73.21	71.52	73.47
GoogLeNet [22]	6.40M	66.20	67.78	77.09	75.67	77.22
DenseNet121 [23]	7.05M	66.02	67.12	77.01	74.62	74.71
SEResNet18 [24]	11.31M	65.64	66.97	74.82	74.94	75.59
Xception [25]	21.01M	66.51	67.45	75.81	75.35	75.55
Inceptionv3 [26]	22.32M	64.69	65.80	75.57	73.78	75.89
VGG11_bn [27]	28.52M	60.88	61.14	68.94	69.94	71.14

Table 2: Maximal test accuracy on CIFAR-100 using regular SGD training and P-SGD training in 40-dimensional subspaces.

For the design of cost-efficient optimization algorithms, it is key to minimize the computational effort spent on sampling.

(a) SqueezeNet

(b) GoogLeNet

Figure 6: Training performance on CIFAR-100 using regular SGD training and P-SGD training in 40-dimensional subspaces.

The complexity of generating samples far exceeds the complexity of performing PCA on these samples.

model	#params	training time per epoch	PCA time for #samples 50	100	150	200
SqueezeNet [18]	0.78M	16.830	0.016	0.020	0.023	0.043
ShuffleNetv2 [19]	1.36M	20.707	0.012	0.020	0.018	0.022
MobileNet [20]	3.32M	16.237	0.020	0.023	0.030	0.104
EfficientNet [21]	4.14M	23.132	0.018	0.028	0.051	0.190
GoogLeNet [22]	6.40M	28.068	0.025	0.030	0.048	0.051
DenseNet121 [23]	7.05M	43.932	0.027	0.033	0.245	0.226
SEResNet18 [24]	11.31M	18.377	0.027	0.250	0.219	0.081
Xception [25]	21.01M	31.819	0.412	0.759	0.473	0.634
Inceptionv3 [26]	22.32M	68.253	0.050	0.584	0.513	0.677
VGG11_bn [27]	28.52M	14.528	0.049	0.076	0.647	0.654

Table 3: DLDR time consumption in seconds (s) to extract 40-dimensional subspaces for training on CIFAR-100.

Extracting low-dimensional subspaces is associated with two problems

Which strategy should be used for sampling?

Which subspace dimension should be used for training?

Samples should be generated by taking the average over the parameters after each batch.

model	Discrete		Average				
	#sampling epochs		#sampling epochs				
	50	100	50	75	100	125	150
SqueezeNet [18]	69.45	71.70	70.19	71.55	71.99	72.36	72.52
ShuffleNetv2 [19]	72.13	73.54	72.27	72.98	73.42	73.33	73.35
MobileNet [20]	67.01	68.76	67.81	68.60	68.63	68.89	68.92
EfficientNet [21]	71.52	73.47	72.15	72.90	73.66	74.05	74.15
GoogLeNet [22]	75.67	77.22	75.85	76.65	77.16	77.19	77.13
DenseNet121 [23]	74.62	74.71	74.54	74.55	74.48	74.52	74.49
SEResNet18 [24]	74.94	75.59	74.99	75.58	75.59	75.70	75.79
Xception [25]	75.35	75.55	75.66	75.54	75.59	75.64	75.58
Inceptionv3 [26]	73.78	75.89	74.72	75.88	75.98	75.91	75.86
VGG11_bn [27]	69.94	71.14	69.98	70.91	71.22	71.21	71.24

Table 4: Maximal test accuracy on CIFAR-100 using 40 epochs of P-SGD training in 40-dimensional subspaces.

Samples should be generated once per epoch and the dimension can be chosen arbitrarily, as long as it is not too small.

#samples	dimension of subspace 2	5	15	25	50	100	500	1000	best
30	50.15	85.44	85.22	85.16	-	-	-	-	85.44
90	63.71	85.42	85.27	85.15	85.10	-	-	-	85.42
150	69.48	85.43	85.27	85.13	85.08	85.14	-	-	85.43
300	76.16	85.47	85.27	85.15	85.14	85.14	-	-	85.47
900	83.41	85.45	85.29	85.12	85.20	85.09	85.14	-	85.45
1500	85.43	85.44	85.33	85.14	85.20	85.15	85.28	85.42	85.44
best	85.43	85.47	85.33	85.16	85.20	85.15	85.28	85.42	85.47

Table 5: Maximal test accuracy of ResNet8 trained on CIFAR-10 using P-SGD in subspaces extracted with average samples.

Possibilties for further work could include the design of cost-efficient low-dimensional training algorithms.

- A future low-dimensional training algorithm could proceed as follows:
  1. Generate samples by training in the full parameter space;
  2. While sampling, decide once enough examples have been generated;
  3. Extract a low-dimensional subspace based on these samples;
  4. Finish off the training by some final epochs of P-SGD in that space;
- **Major problem to solve:** Finding a criterium to decide once enough examples have been generated.

References

[1] Tao Li, Lei Tan, Qinghua Tao, Yipeng Liu, Xiaolin Huang. *Low Dimensional Trajectory Hypothesis is True: DNNs can be Trained in Tiny Subspaces*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

[2] Arthur Jacot, Franck Gabriel, Clément Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. Advances in Neural Information Processing Systems, volume 31, pages 8571-8580, 2018.

[3] Zhou Fan, Zhichao Wang. *Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks*. Advances in Neural Information Processing Systems, volume 33, pages 7710-7721, 2020.

[4] Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, Ronen Basri. *On the Similarity between the Laplace and Neural Tangent Kernels*. Advances in Neural Information Processing Systems, volume 33, pages 1451-1461, 2020.

Are there any questions?