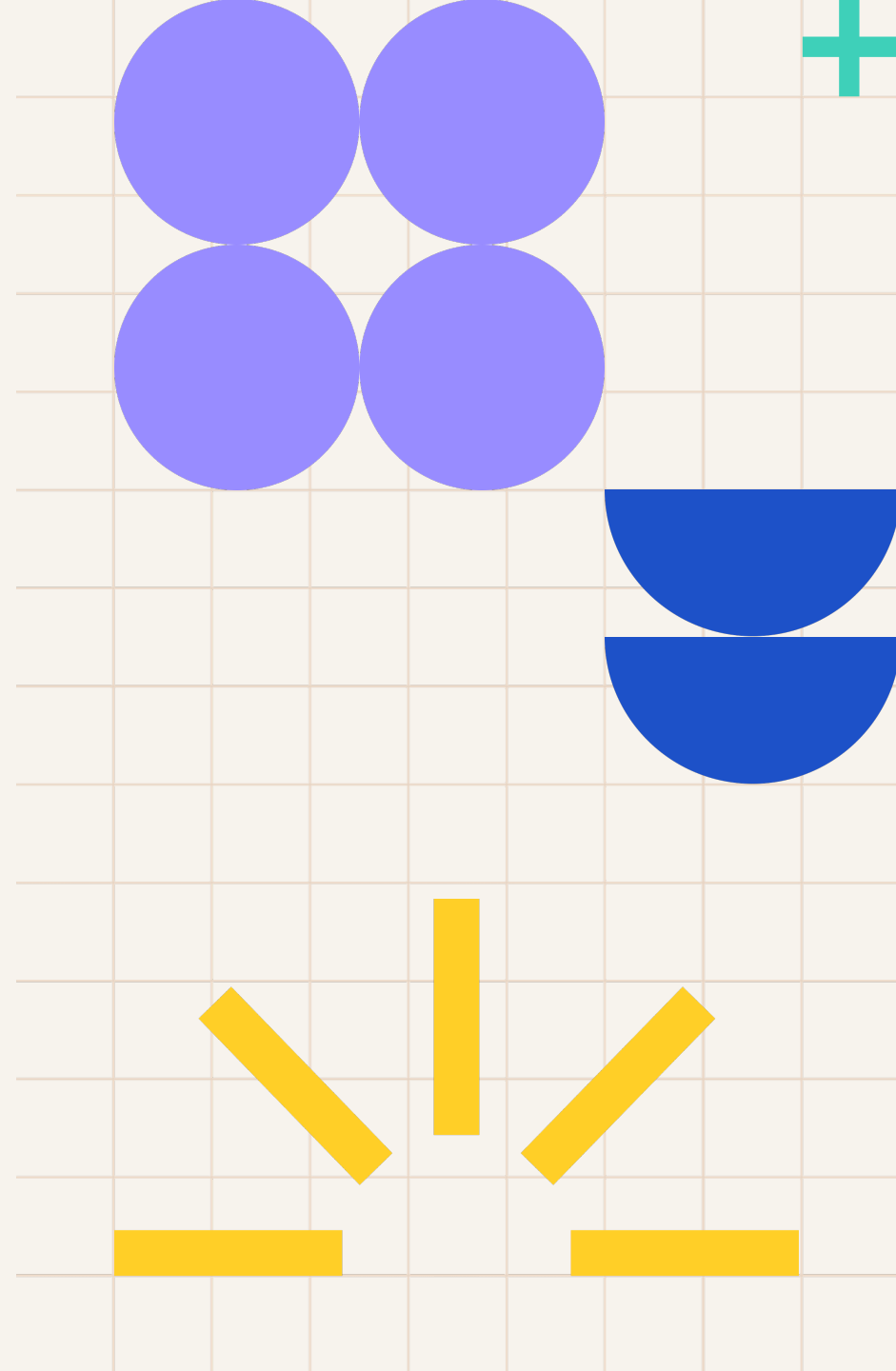




# Optimisation methods for training large models

Aleksandr Katrutsa

Skoltech, AIRI



# Plan

---

- 01 Problem statement
- 02 Basic optimisers
- 03 What is changed with large models?
- 04 Main approaches
- 05 Summary



# Problem statement



# What problem we are going to solve?

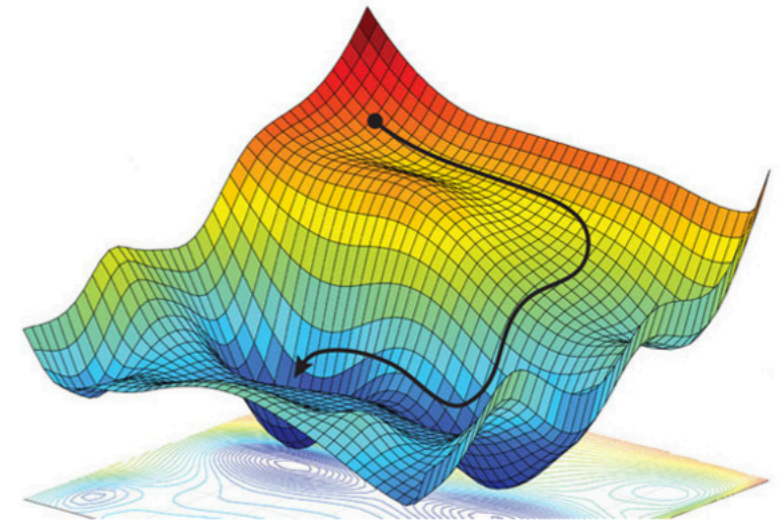
- Given dataset  $X = \{x_i\}_{i=1}^N$
- According to the target task we have the corresponding loss function

$$L(w | X) = \frac{1}{N} \sum_{i=1}^N L(w | x_i), \quad L_i(w) \equiv L(w | x_i)$$

and the model  $f(w, x_i)$  which is simulated by DNN

- The main problem is

$$w^* = \arg \min_w L(w | X)$$



Source is [here](#)

## Main features of the problem

- Non-convexity: multiple local minima of different quality
- Overfitting is a problem although DNNs are typically overparametrized
- Generalisation is important property of the final model
- Adversarial attacks are also a challenge in model training
- DNNs require massive parallel computations to treat the sufficient amount of data

---

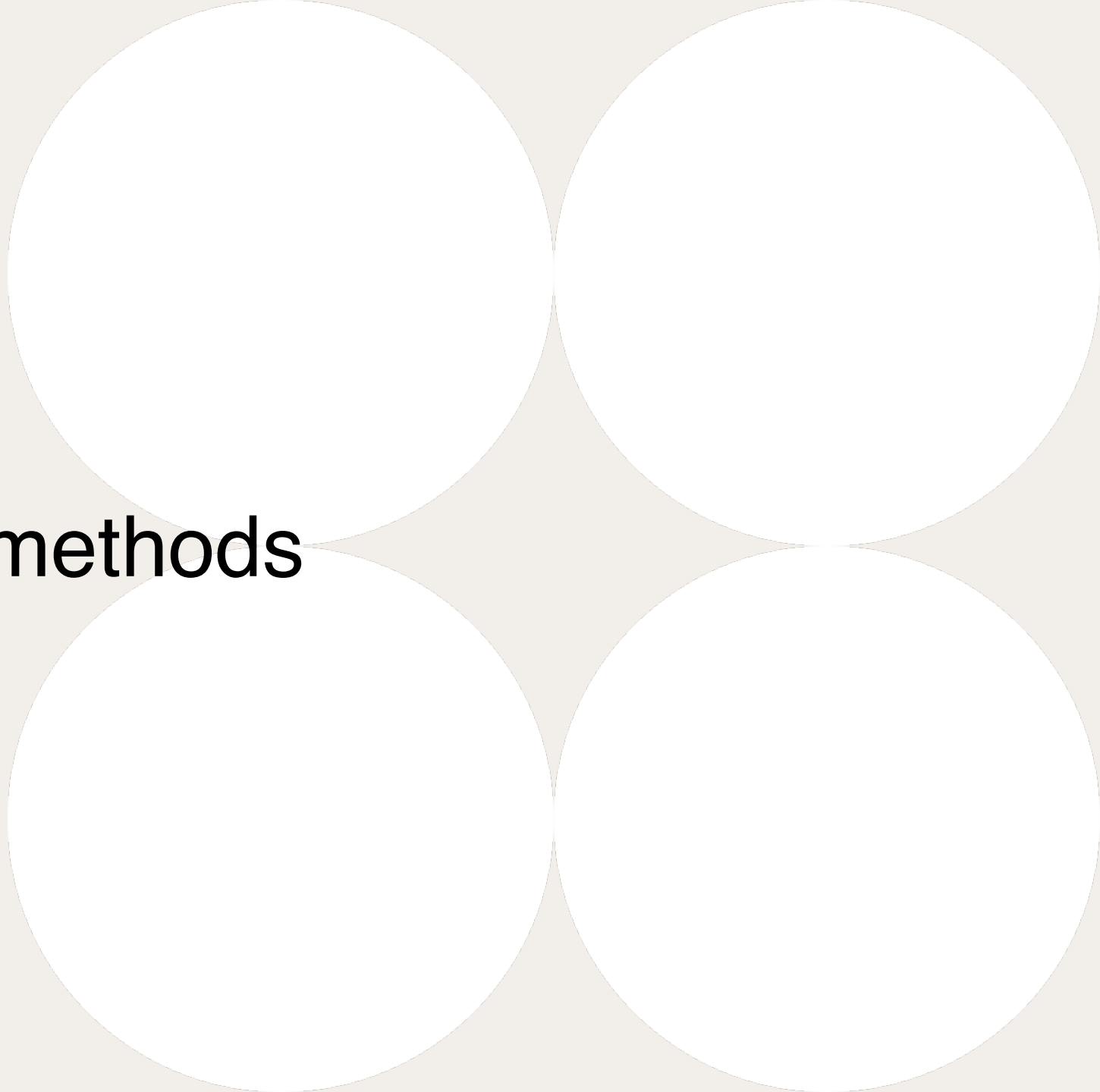
## Features of the problem statement:

- Non-convexity
- Non-smoothness
- Generalisation
- Robustness w.r.t. adversarial attacks
- Limited of the computational and storage resources

# 02

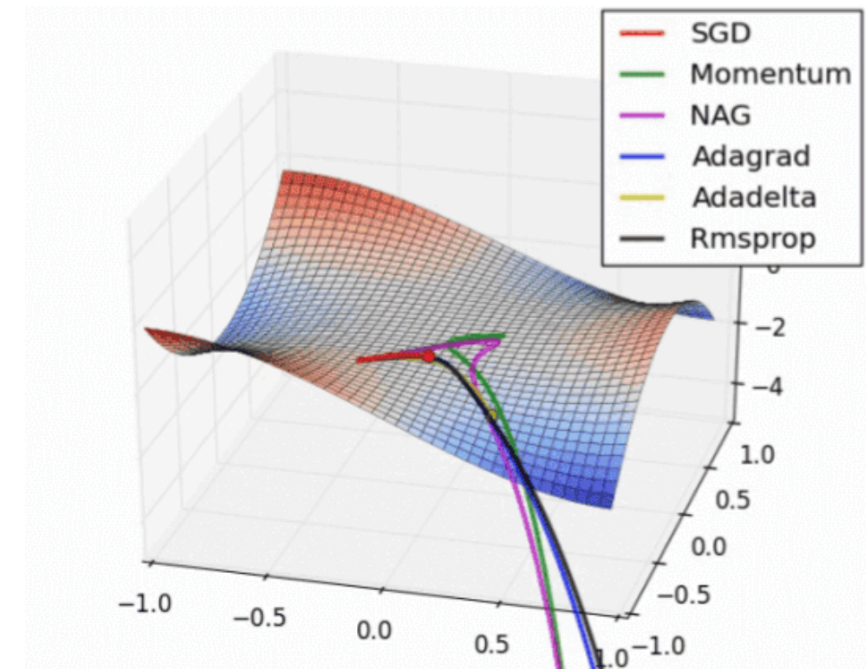


## Basic optimisation methods



## What do we want from optimisers?

- Train loss minimisation
- High generalisation, i.e. test loss minimisation simultaneously
- Fast convergence
- No special tuning of hyperparameters
- Adversarial robustness to attacks of the resulting model
- Low consumption of the additional memory



Source is [here](#)

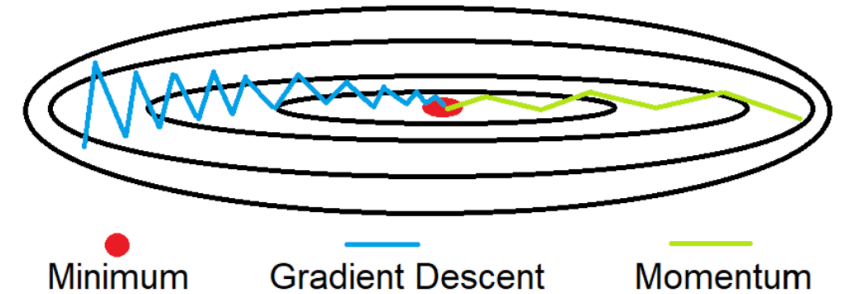


# Stochastic gradient descent

- Gradient estimation from batch  $B_k$

$$L'_k(w) = \frac{1}{|B_k|} \sum_{i \in B_k} L'_i(w)$$

- SGD:  $w_{k+1} = w_k - \alpha L'_k(w)$ ,  $\alpha > 0$
- SGD + Momentum:  $w_{k+1} = w_k - \alpha L'_k(w) + \beta(w_k - w_{k-1})$
- Adaptive learning rate:  $\alpha = \text{const} \rightarrow \alpha_k = s(\cdot)$



Source is [here](#)

What we can already observe?

- Batch size  $|B_k|$  affects the noise in gradient estimate
- Momentum methods require storage additional vectors
- Learning rate tuning can help

## Some theory about such methods

### Theorem

Let  $f$  be convex,  $L$ -smooth function. Then if SGD generates directions  $\mathbf{h}_k$  such that  $\text{Var}(\mathbf{h}_k) \leq \sigma^2$  and  $\alpha_k \leq \frac{1}{L}$  then

$$\mathbb{E}[f(\bar{\mathbf{x}}_k)] - f^* \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{\alpha_k k} + \frac{\alpha_k \sigma^2}{2}.$$

In particular, after  $k = \frac{(\sigma^2 + L\|\mathbf{x}^* - \mathbf{x}_0\|_2^2)^2}{\varepsilon^2}$  iterations if  $\alpha_k = \frac{1}{\sqrt{k}}$  we get the solution with accuracy  $2\varepsilon$ .

- No convergence due to noise term
- Averaging of gradients leads to convergent method - SAG [Schmidt, et al, 2013] - impractical due to memory limitations
- Only for convex case the theory is well-developed

# Adaptive learning rate methods

- Adam [Kingma, et al, 2014] is the most representative method

$$w_{k+1} = w_k - \alpha \frac{\hat{m}_k}{\sqrt{\hat{v}_k + \epsilon}}$$

- $\hat{m}_k = \frac{m_k}{1 - \beta_1^k}$ ,  $m_k = \beta_1 m_{k-1} + (1 - \beta_1) L'_k$
- $\hat{v}_k = \frac{v_k}{1 - \beta_2^k}$ ,  $v_k = \beta_2 v_{k-1} + (1 - \beta_2) (L'_k \cdot L'_k)$
- All operations are elementwise

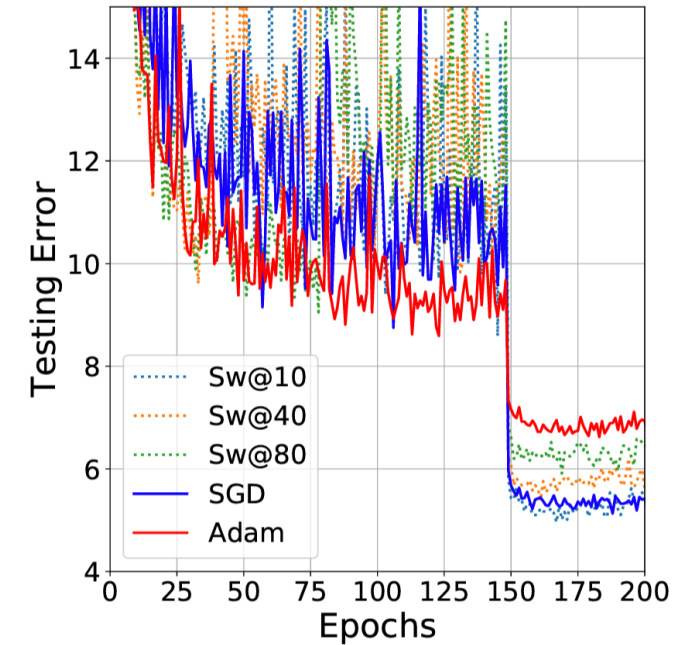
## SGD vs Adam

- Adam stores two vectors of the size of #parameters
- More computations per step
- Learning rate in Adam differs from SGD

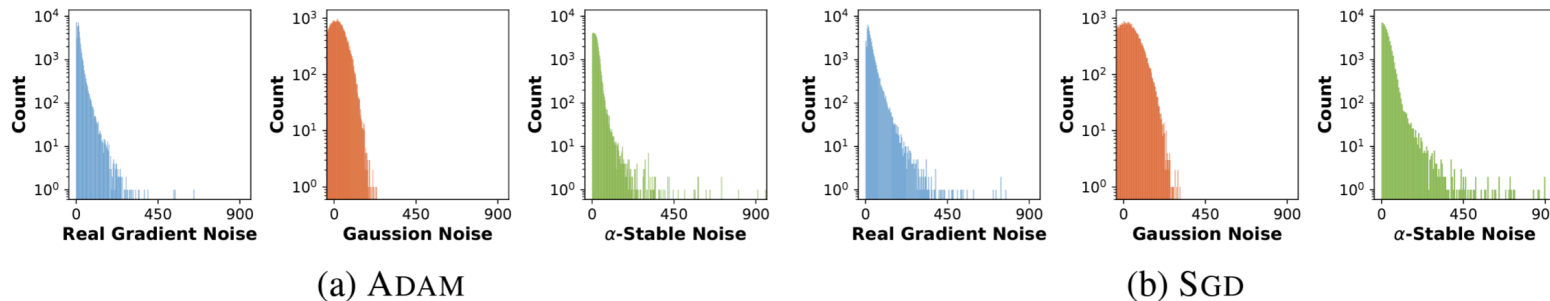
# Adam can generalise worse

- It was observed that though Adam converges faster, it gives less generalisable model
- Possible explanations

- SGD faster escapes from the local minima
- Adam average leads to lighter gradient noise tails
- Formal proofs are based on SDE interpretation of SGD:  
$$dw_t = -L'(w_t)dt + \varepsilon \Sigma_t dL_t^{(s)}$$
, where  $L_t$  is Levi process



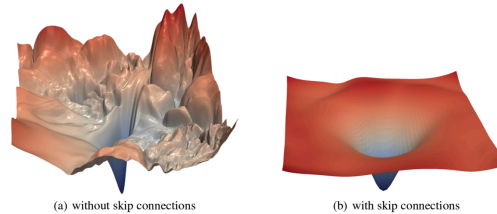
Source is ["Improving Generalization Performance by Switching from Adam to SGD"](#), N.S.Keskar et al, 2017



Source is ["Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning"](#), P. Zhou et al, NeurIPS 2020

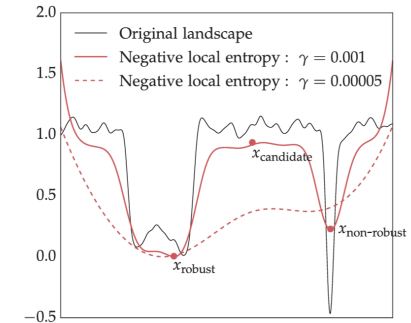
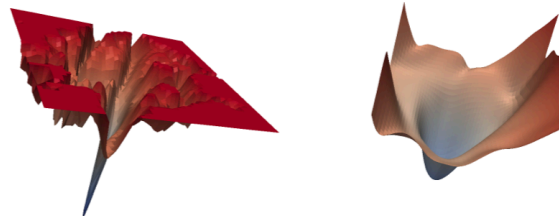
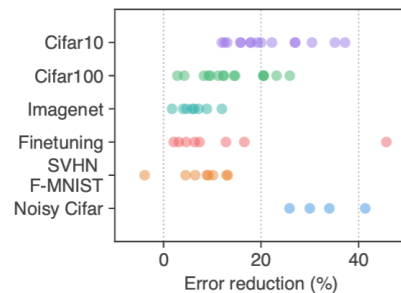
# Flat vs sharp local minima

- Generalisation is also affected by the geometry of loss landscape
- Compare different architectures

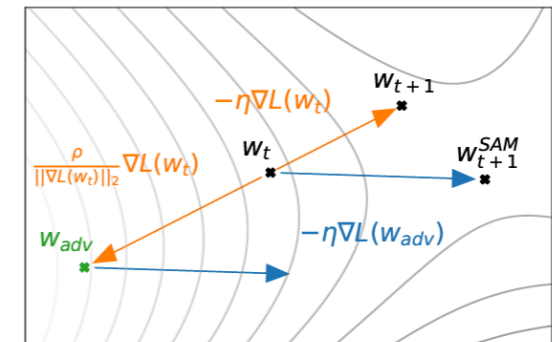


ResNet-56 architecture. Source is [“Visualizing the Loss Landscape of Neural Nets”, H. Li et al, NeurIPS 2018](#)

- Sharpness aware minimisation (SAM) method suggests computing adversarial direction and updates parameters to avoid










Source is “Entropy-SGD: biasing gradient descent into wide valleys”, Pratik Chaudhari et al J. Stat. Mech. (2019)



Source is [“Sharpness-aware minimization for efficiently improving generalization”, P. Foret, ICLR 2020](#)

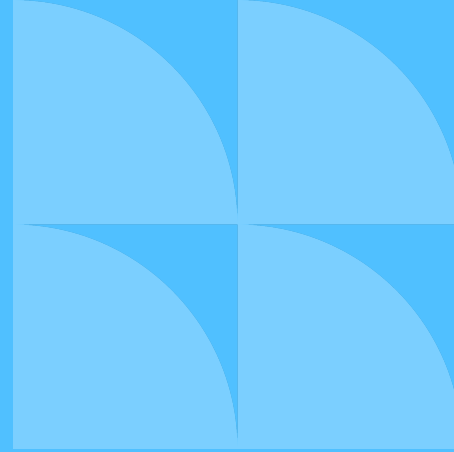
# Statistics from <https://paperswithcode.com/>

Method	Year	Papers
 <b>Adam</b> Adam: A Method for Stochastic Optimization	2014	8544
 <b>SGD</b>	1951	1286
 <b>RMSProp</b>	2013	316
 <b>Adafactor</b> Adafactor: Adaptive Learning Rates with Sublinear Memory Cost	2018	220
 <b>LAMB</b> Large Batch Optimization for Deep Learning: Training BERT in 76 minutes	2019	138
 <b>SGD with Momentum</b>	1999	133
 <b>AdaGrad</b>	2011	112

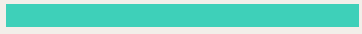
---

# Main features of basic optimisers

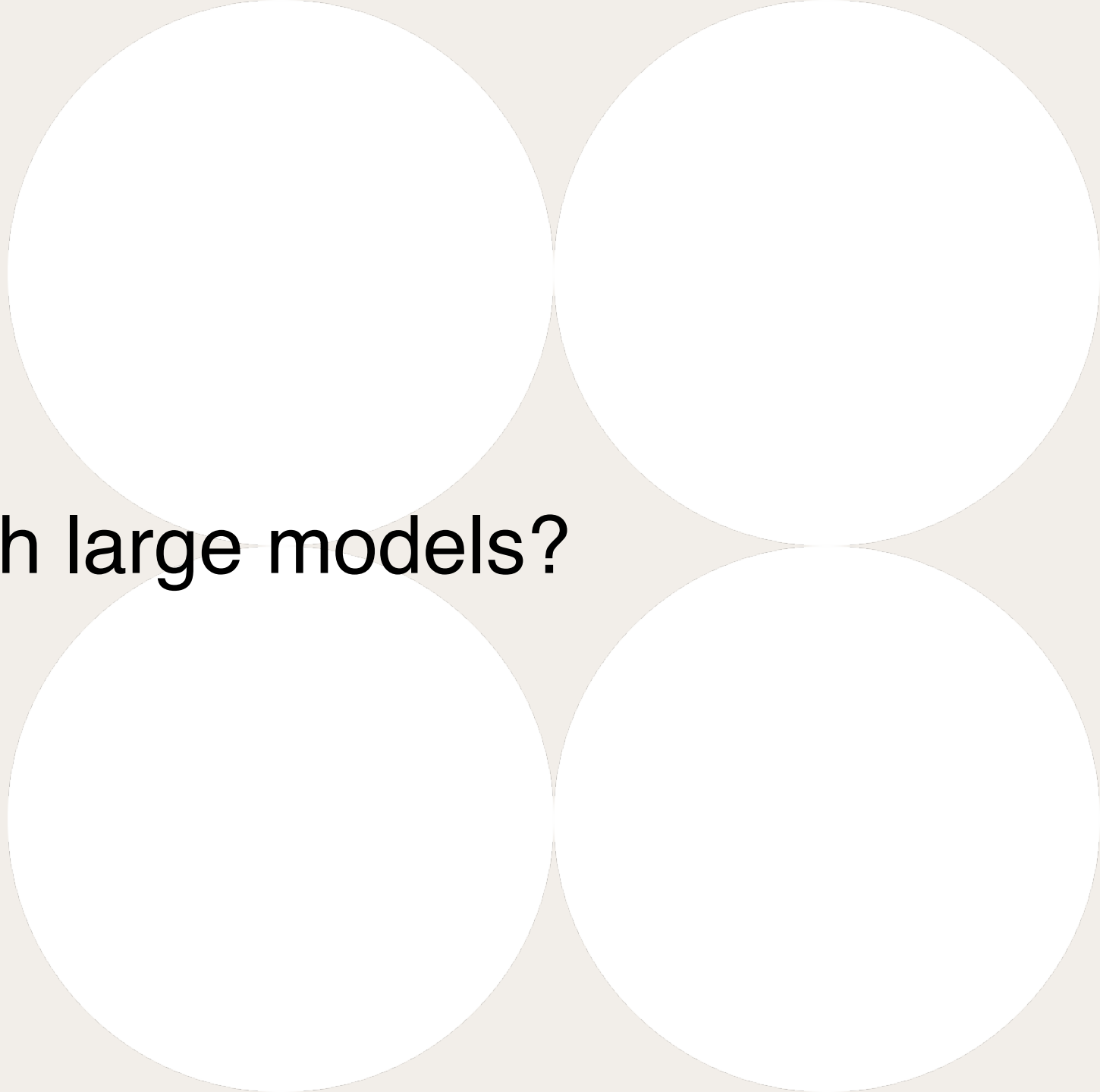
- Adaptive learning rate
- Smoothing of gradient statistics



# 03



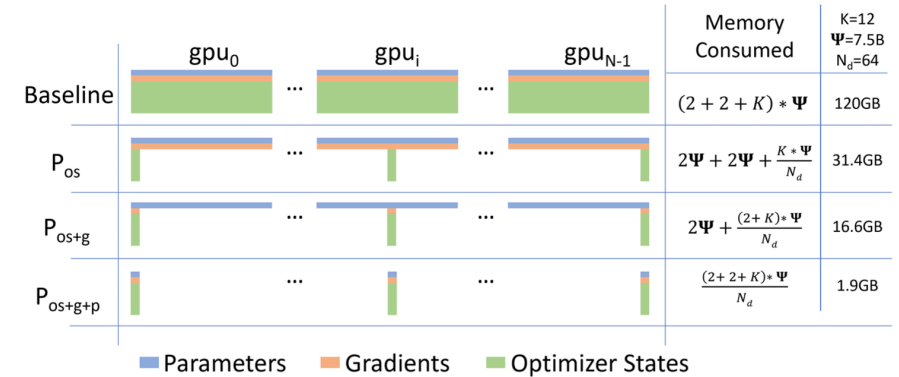
What is changed with large models?





# Large models from the optimisation perspective

- Huge #parameters affects the storage of optimiser state
- Parameters of the model can be stored in distributed manner. Remember about ZeRO framework
- The communication between GPUs appears - new factor in our scheme
- Large batch size increases GPU utilisation but affects the scheduler of other hyperparameters



Source is [here](#)

---

## New factors

- We need more memory: CPU vs GPU
- Communications between nodes
- Distributed storage of model and optimiser state

# 04

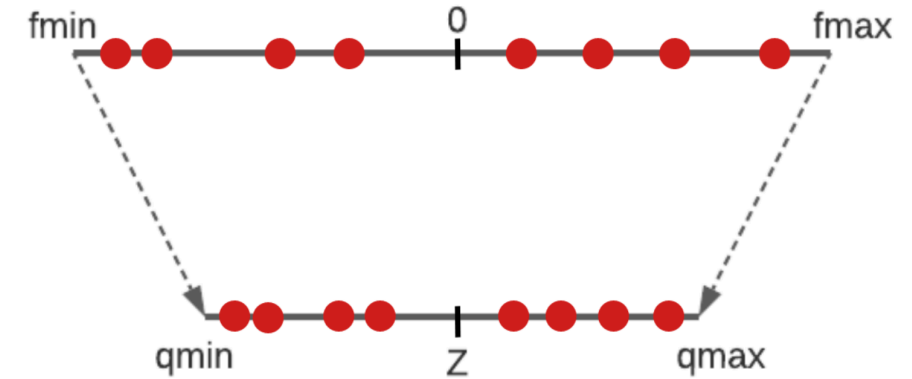


## Main approaches

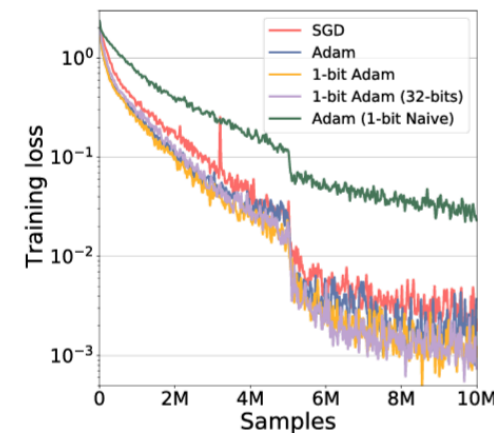


# Quantisation

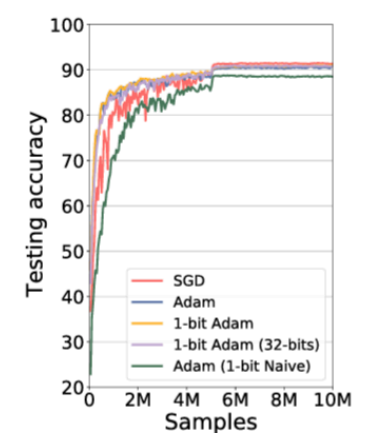
- The easy way to reduce memory footprint - decrease the accuracy of numbers representation
- It affects the property of optimisers
- FP32 is standard format
- [FP16](#), [bfloat](#) and [TensorFloat](#) alternatives
- Even just integers can be used [S. Kim et al, 2021]
- Quantisation is for optimiser state or for parameters
- Extreme case: 1-bit SGD and 1-bit Adam for efficient communications



Source is [here](#)



(a) Training loss

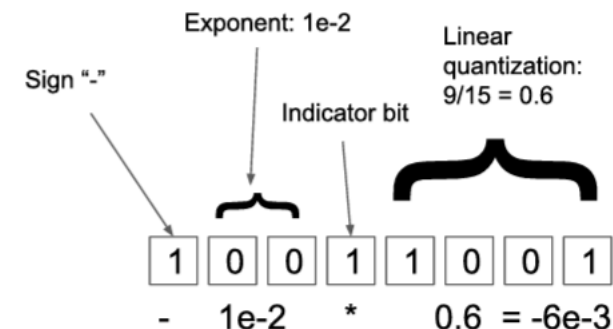


(b) Testing accuracy

ResNet-18, CIFAR-10, source is [here](#)

# Adaptive quantisation, 8-bit Adam

- Blockwise quantisation improves optimisers
- Example is [here](#) for momentum and Adam
- Idea: adaptively choose the range of values that should be covered by the quantised numbers



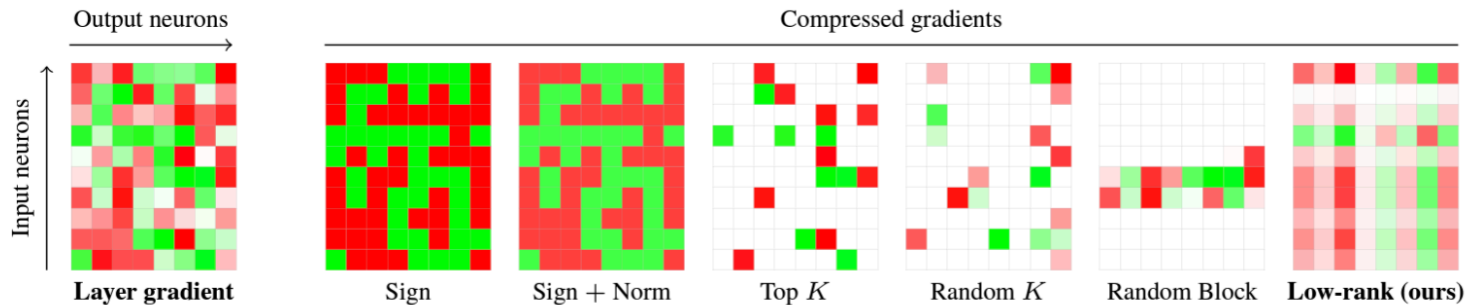
Optimizer	Task	Data	Model	Metric <sup>†</sup>	Time	Mem saved
32-bit AdamW	GLUE	Multiple	RoBERTa-Large	88.9	–	Reference
32-bit AdamW	GLUE	Multiple	RoBERTa-Large	88.6	17h	0.0 GB
32-bit Adafactor	GLUE	Multiple	RoBERTa-Large	<b>88.7</b>	24h	1.3 GB
8-bit AdamW	GLUE	Multiple	RoBERTa-Large	<b>88.7</b>	<b>15h</b>	<b>2.0 GB</b>
32-bit Momentum	CLS	ImageNet-1k	ResNet-50	77.1	–	Reference
32-bit Momentum	CLS	ImageNet-1k	ResNet-50	77.1	118h	0.0 GB
8-bit Momentum	CLS	ImageNet-1k	ResNet-50	<b>77.2</b>	<b>116 h</b>	<b>0.1 GB</b>
32-bit Adam	MT	WMT' 14+16	Transformer	29.3	–	Reference
32-bit Adam	MT	WMT' 14+16	Transformer	29.0	126h	0.0 GB
32-bit Adafactor	MT	WMT' 14+16	Transformer	29.0	127h	0.3 GB
8-bit Adam	MT	WMT' 14+16	Transformer	<b>29.1</b>	<b>115h</b>	<b>1.1 GB</b>
32-bit Momentum	MoCo v2	ImageNet-1k	ResNet-50	67.5	–	Reference
32-bit Momentum	MoCo v2	ImageNet-1k	ResNet-50	67.3	30 days	0.0 GB
8-bit Momentum	MoCo v2	ImageNet-1k	ResNet-50	<b>67.4</b>	<b>28 days</b>	<b>0.1 GB</b>
32-bit Adam	LM	Multiple	Transformer-1.5B	9.0	308 days	0.0 GB
32-bit Adafactor	LM	Multiple	Transformer-1.5B	<b>8.9</b>	316 days	5.6 GB
8-bit Adam	LM	Multiple	Transformer-1.5B	9.0	<b>297 days</b>	<b>8.5 GB</b>
32-bit Adam	LM	Multiple	GPT3-Medium	10.62	795 days	0.0 GB
32-bit Adafactor	LM	Multiple	GPT3-Medium	10.68	816 days	1.5 GB
8-bit Adam	LM	Multiple	GPT3-Medium	<b>10.62</b>	<b>761 days</b>	<b>1.7 GB</b>
32-bit Adam	Masked-LM	Multiple	RoBERTa-Base	3.49	101 days	0.0 GB
32-bit Adafactor	Masked-LM	Multiple	RoBERTa-Base	3.59	112 days	0.7 GB
8-bit Adam	Masked-LM	Multiple	RoBERTa-Base	<b>3.48</b>	<b>94 days</b>	<b>1.1 GB</b>

<sup>†</sup>Metric: GLUE=Mean Accuracy/Correlation. CLS/MoCo = Accuracy. MT=BLEU. LM=Perplexity.

GPU size in GB	Largest finetunable Model (parameters)	
	32-bit Adam	8-bit Adam
6	RoBERTa-base (110M)	RoBERTa-large (355M)
11	MT5-small (300M)	MT5-base (580M)
24	MT5-base (580M)	MT5-large (1.2B)
24	GPT-2-medium (762M)	GPT-2-large (1.5B)

# Communications costs reduction

- Different methods for compression: algorithmic and network
- PowerSGD: low-rank compression from the power method
- Alternative compression methods
  - Sign
  - Sign + norm
  - Top-K
  - Random
  - Block random
- Zero-Infinity: proper memory and network usage



Source is [here](#)

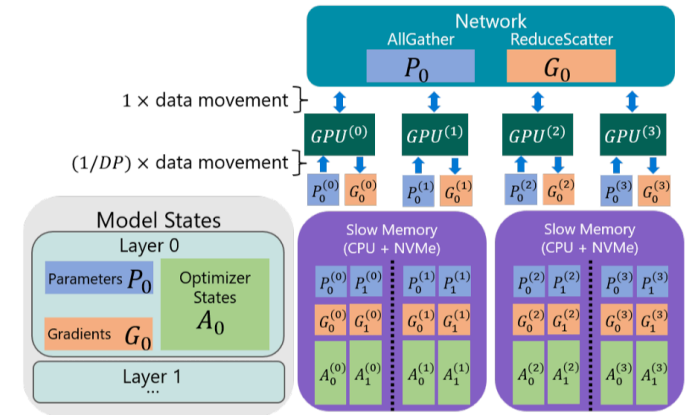


Figure 4: A snapshot of ZeRO-Infinity training a model with two layers on four data parallel (DP) ranks. Communication for the backward pass of the first layer is depicted. Partitioned parameters are moved from slow memory to GPU and then collected to form the full layer. After gradients are computed, they are aggregated, re-partitioned, and then offloaded to slow memory. Layers are denoted with subscripts and DP ranks are denoted with superscripts. For example,  $P_0^{(2)}$  is the portion of layer 0's parameters owned by  $GPU^{(2)}$ .

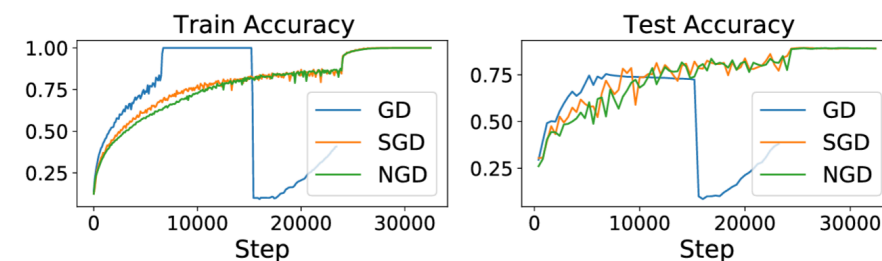
Source is [here](#)

# Large batch training: idea

- Increase batch to utilise GPUs better
- This modification affects the gradient estimate noise
- As a result some modification of learning rate is needed to preserve learning curve
- Analysis of required transformation is easy with SDE approximation

$$dw_t = -L'(w_t)dt + (\alpha\Sigma(w_t))^{1/2}dW_t,$$

where  $W_t$  is Wiener process

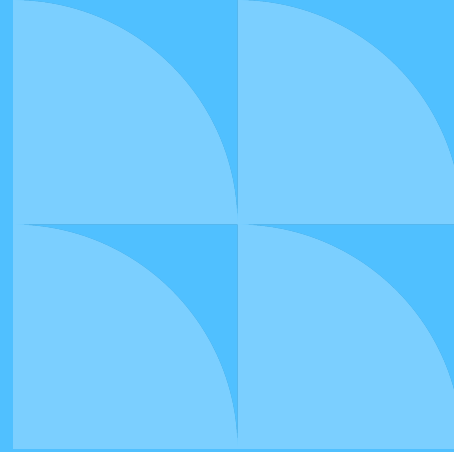


Source is [“On the Validity of Modeling SGD with Stochastic Differential Equations \(SDEs\)”](#), Z. Li et al. NeurIPS 2021

---

# Main recipes

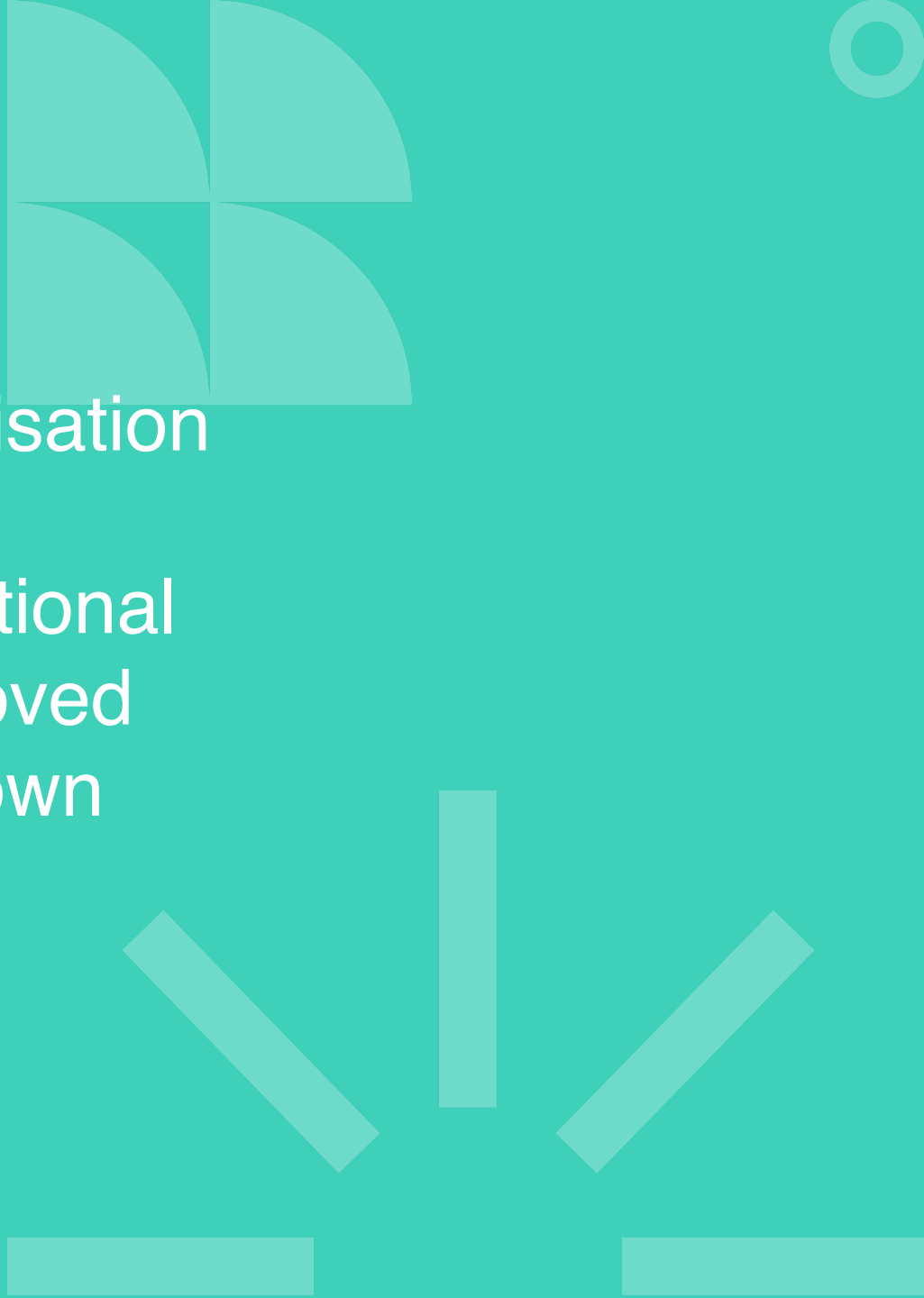
- Quantisation
- Compression
- Maximum GPUs utility





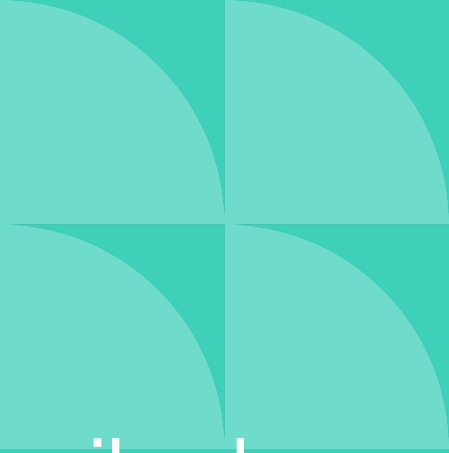




# Summary

- Large models require special optimisation techniques
  - Existing approaches are multi-directional
  - Every direction can be further improved since no theoretical bounds are known
- 



# Questions

- What is a proper combination of described approaches?
  - How test them fairly?
  - Is it possible to formal prove the described heuristics?
- 
- 
- 



# Artificial Intelligence Research Institute

airi.net



[airi\\_research\\_institute](https://t.me/airi_research_institute)



[AIRI Institute](https://vk.com/AIRI_Institute)



[AIRI Institute](https://www.youtube.com/AIRI_Institute)



[AIRI\\_inst](https://twitter.com/AIRI_inst)



[artificial-intelligence-research-institute](https://www.linkedin.com/company/artificial-intelligence-research-institute)