


Postdoktor Vladislav Mikhailov, førsteamanuensis Andrei Kutuzov, PhD-stipendiat David Samuel og professor Erik Velldal, alle fra forskningsgruppen for språkteknologi på Institutt for informatikk ved UiO. Foto: Gina Aakre

# Store skritt mot norsk svar på ChatGPT

Flere nye norske språkmodeller er allerede lansert.

– Nasjonalbiblioteket og Universitetet i Oslo har tilgjengeliggjort flere norske språkmodeller tidligere, men dette er de største vi har laget så langt og de er trent på over 30 milliarder ord, forklarer Velldal.

Alle modellene har rundt syv milliarder parametre, noe som forskerne vurderer som optimalt i forhold til mengden norske treningsdata som er tilgjengelig.

– En språkmodell blir dårlig om den trenes på for lite materiale i forhold til størrelsen. Det handler om å finne rett balanse, sier Velldal.

Et triks som har vist seg effektivt dersom man har nok prosessorkraft er å trene på de samme dataene i flere runder. Modellene UiO har trent på LUMI har blitt matet med de samme treningsdataene seks ganger, forklarer Velldal.

oktober i det nasjonale
NORA planlegger partn
infrastruktur for utviklir
norske språkmodeller.

To av modellene er tren
norsk.

Den tredje tar utgangsp
forhåndstrent for Enge
selskapet Mistral AI, so
videre for norsk.

Modellene er tilgjengel
https://huggingface.co

- This week's lecture recommends 10x more training data than ANN parameters. Your language model has 7 billion parameters and 30 billion words. **How do you avoid overfitting?**
- **Why do you train on the same data 6 times? What would happen if you train on them even more times?**

– Nasjonalbiblioteket og Universitetet i Oslo har tilgjengeliggjort flere norske språkmodeller tidligere, men dette er de største vi har laget så langt og de er trent på over 30 milliarder ord, forklarer Velldal.

Alle modellene har rundt syv milliarder parametre, noe som forskerne vurderer som optimalt i forhold til mengden norske treningsdata som er tilgjengelig.

– En språkmodell blir dårlig om den trenes på for lite materiale i forhold til størrelsen. Det handler om å finne rett balanse, sier Velldal.

Et triks som har vist seg effektivt dersom man har nok prosessorkraft er å trene på de samme dataene i flere runder. Modellene UiO har trent på LUMI har blitt matet med de samme treningsdataene seks ganger, forklarer Velldal.

NORA planlegger partr
infrastruktur for utviklir
norske språkmodeller.

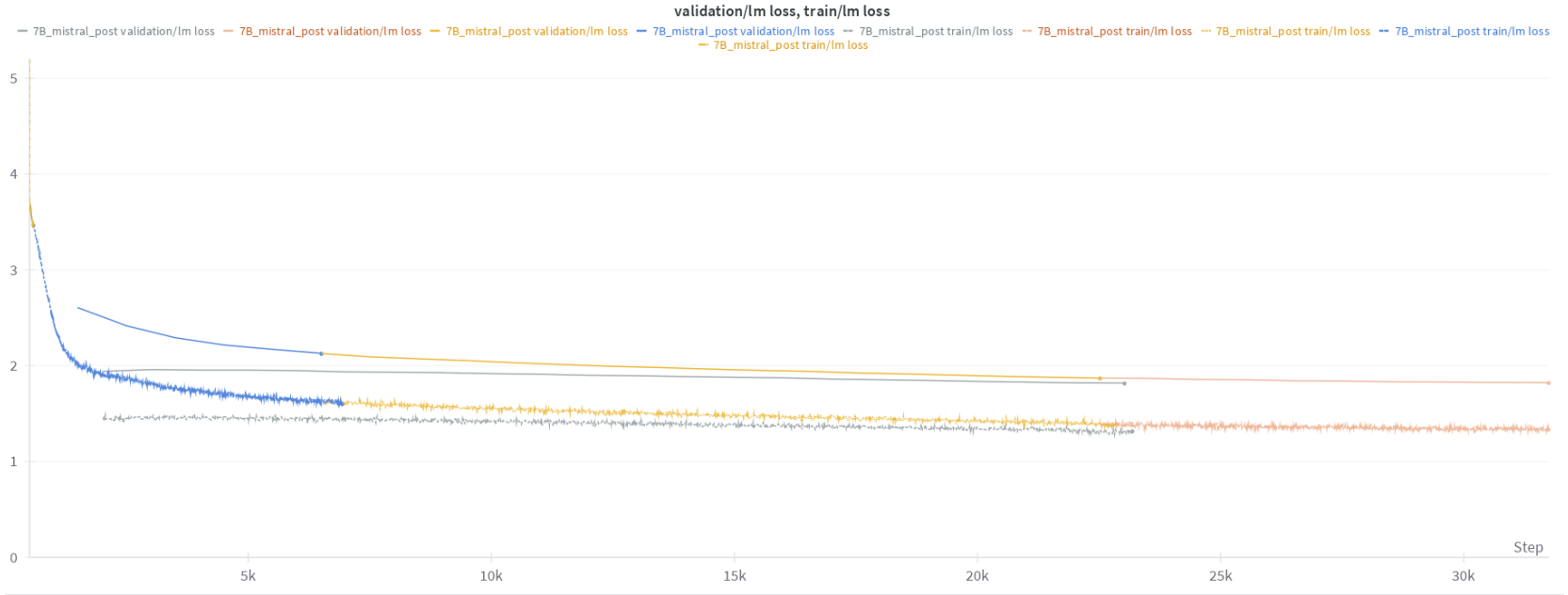
To av modellene er trer
norsk.

Den tredje tar utgangs
forhåndstrent for Enge
selskapet Mistral AI, so
videre for norsk.

Modellene er tilgjengel
https://huggingface.co

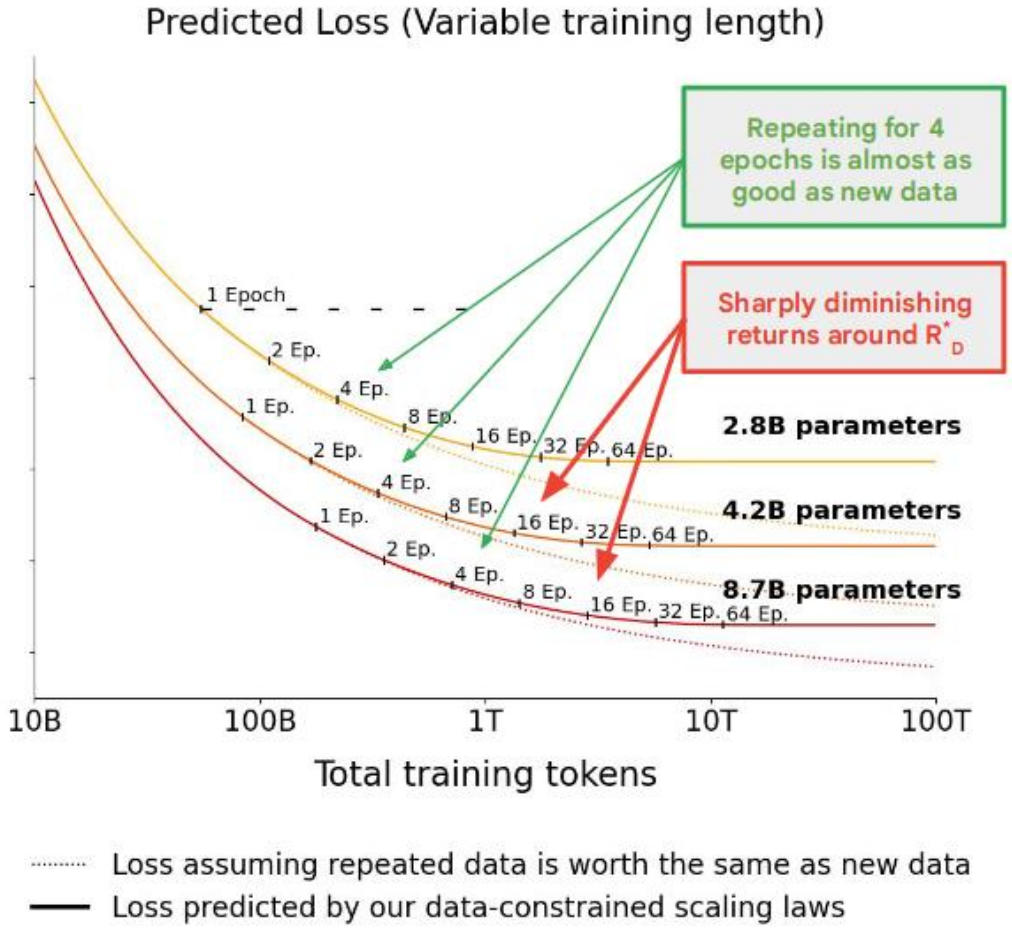
# Large language models training

1. The lecture recommends 10x more training instances than *features*, not than *model parameters/weights* :-)
   - 34B x 6 ~ **200B training instances** (subwords) in NORA.LLM models; in fact, even more, because of programming code
   - Difficult to say what exactly is a feature in an LLM. But even if every model parameter is a separate feature, we are still well above 70B training instances (7B x 10). Far from over-fitting:



# Large language models training

1. Data repetition is needed because we are in a **data-constrained regime:**
   - See "Training compute-optimal large language models" by Hoffmann et al, 2022
   - Its **"scaling laws"** tell us there's not enough Norwegian texts to train a 7B model in the **infinite data regime** (using only unique data points)
   - A 7B model would be under-trained if seeing only 34B sub-words
     - Much compute power basically wasted in vain
     - Would be more efficient and environmentally friendly to train a 1.5B model then
   - But! See also "Scaling data-constrained language models" by Muennighoff et al, 2023
   - 4-6 repetitions is almost the same as new data
     - (quick degradation after that)
   - A way to train larger/better compute-optimal LLMs than the existing data allows



# Making your own LLM

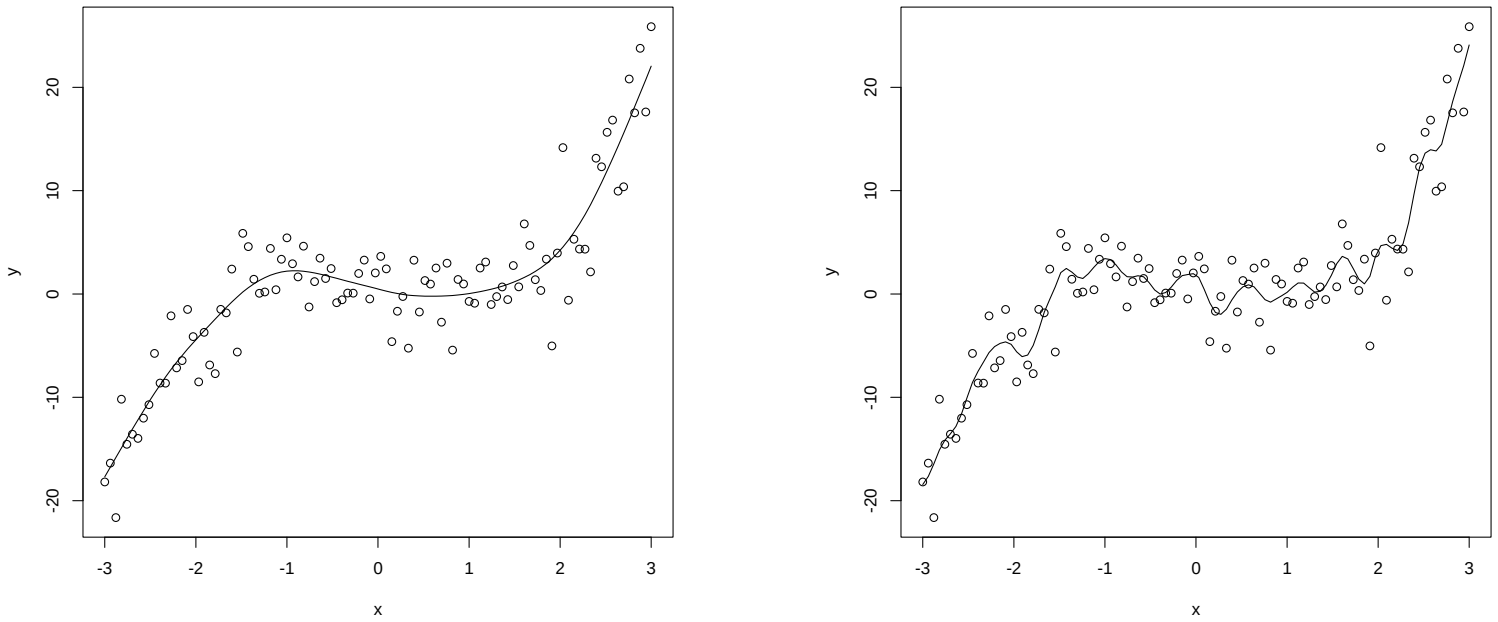
- How many people do you need to do this kind of work?
  - I assume OpenAI had hundreds of people working on ChatGPT. Seems impressive to do this in a relatively small research group!
- What is the most difficult/time-consuming part?
  - Finding the data?
  - Selecting the model architecture/hyperparameters?
  - Getting the training to run on the supercomputer?
  - Programming?
  - Something else?

# Making your own LLM

- Training is done by code and GPUs, not by people :)
- 2-3 competent NLP researchers are enough, given
  - Enough data
  - Enough compute
- Language Technology Group has good experience with high-performance computing and good allocations on the LUMI supercomputer
- The most difficult part was finding good data (texts) and processing it.
- Humans are still needed for instruction-finetuning, reinforcement learning with human feedback and other "alignment" of an already trained model
- Of course: **evaluation**!

# How to detect overfitting?

## Two curve estimates for the same data set:



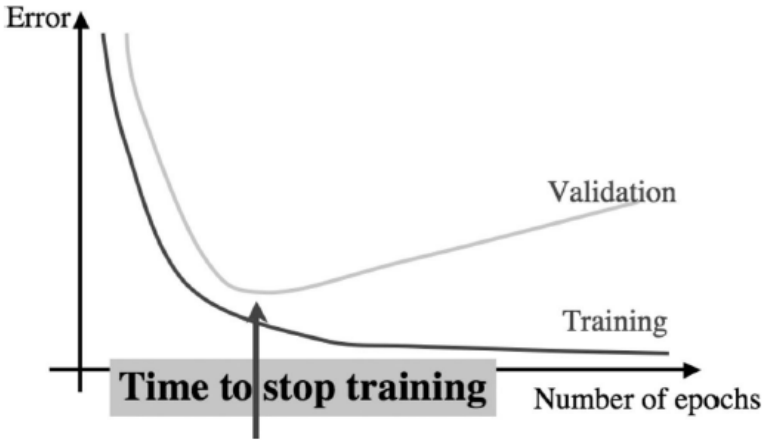




## For this simple 2D problem it is visually clear that the right curve overfits the data.

## Most machine learning problems involve higher-dimensional data. How can we detect overfitting in that case?

# Early stopping




Principle:

Stop the learning when the error starts increasing on the validation data set

Idea:

When the error starts increasing on the validation data, we are likely overfitting to the training data

# Regularization

- The lecture states that regularization reduces overfitting
- But all the regularizer does is punish large weights

- Why do smaller weights overfit less?
- How do I choose between different regularizers (Ridge, Lasso, Elastic Nets)
- And why are they called Ridge, Lasso, Elastic Nets?

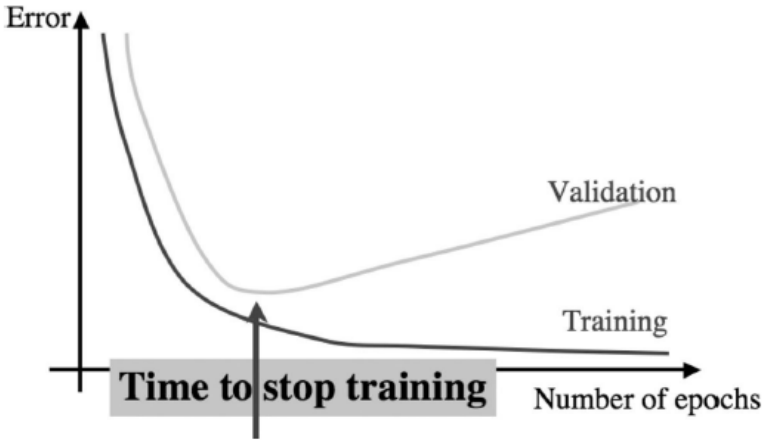
**Regularization**

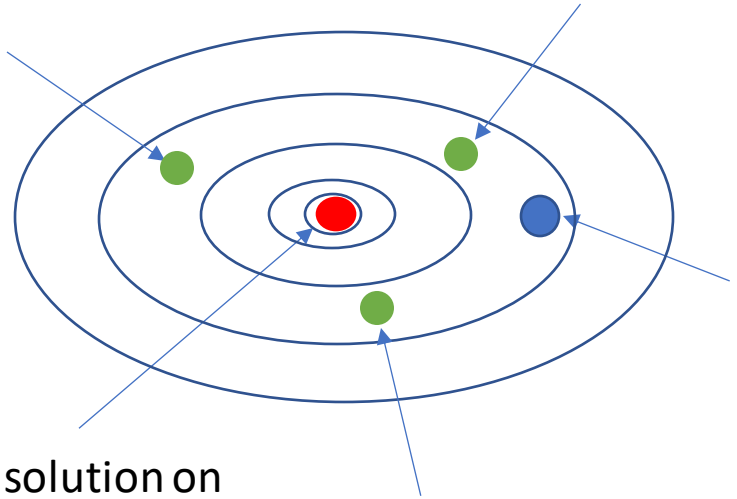
- Replace
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w})$ with
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) - \alpha R(\boldsymbol{w})$
- Where
  - $R(\boldsymbol{w})$ punishes large weights
  - $\alpha$ determines how hard they should be punished
  - $\alpha$ is to be tuned on a validation set

# Regularization

- Large weights = complexity
- Large weights = ability to lower the loss by over-adapting
- We want the model which is as **simple** as possible, even if the loss is slightly higher
  - Because this model will generalize better
  - Also, in many cases, easier to train
- The choice between different regularizers should be informed by experiments, as usual
- Modern deep learning mostly uses **Dropout**.

# Early stopping



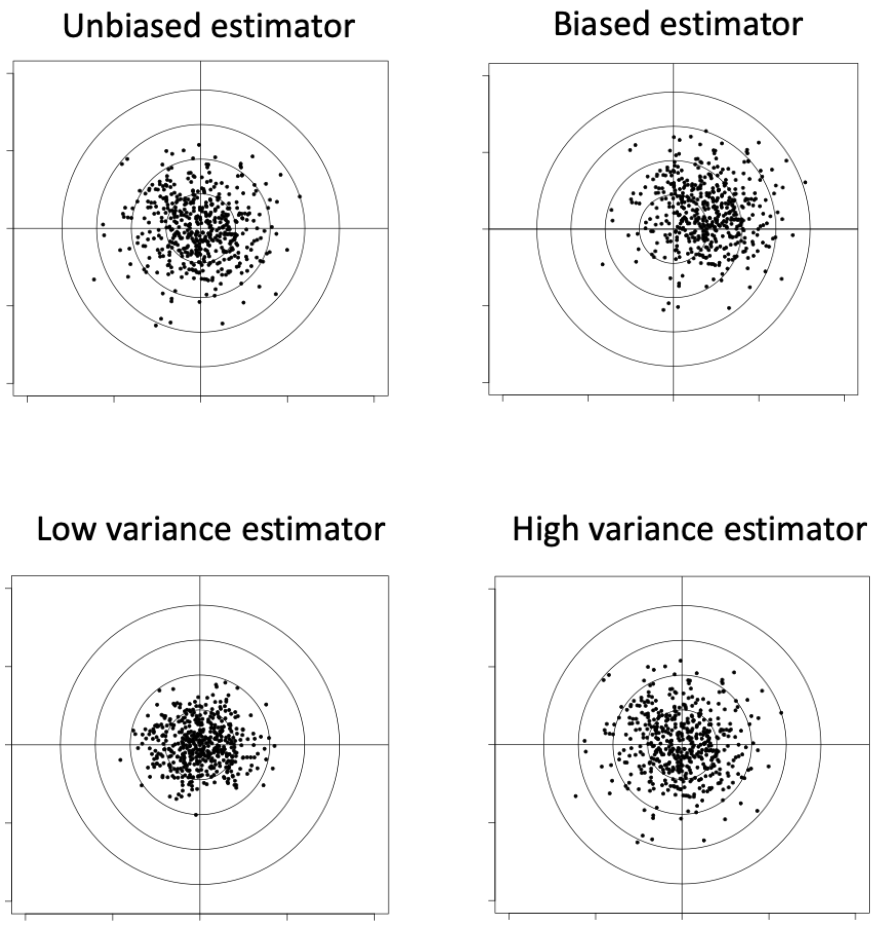



True solution (optimal solution on future data sets)

Optimal solution on training data set

- It seems early stopping is very dependent on the start state?
- Can early stopping give a worse result than not stopping early?

# Bias and variance – what is most important to control?




**High bias means the solutions on different data sets are not centred on the right target (the truth).**

**High variance means the solutions on different data sets have large spread.**

**- What factors influence bias?**

**- What factors influence variance?**

**- Why can't I have low bias and variance at the same time?**

# Ensemble classifiers

**Key idea:**
**- To learn: Train many weak classifiers**
**- To predict: apply all the classifiers and decide the answer by a vote**

**Questions:**

- What makes ensemble classifiers so effective? Any intuition?
- How many weak classifiers do we typically need?
- Would it be better to train many *strong* classifiers?

# Ensemble classifiers

**Questions:**

- What makes ensemble classifiers so effective? Any intuition?
- How many weak classifiers do we typically need?
- Would it be better to train many *strong* classifiers?

**Answers:**

- We minimize the probability of missing an extremely lucky classifier or landing on an extremely unlucky one
- No clear answer, depends on the task
- Of course! But strong classifiers:
  - are more computationally expensive
  - might not be possible to train at all

# Pros and cons of ANNs

**The course introduces different machine learning methods, such as:**

- k-nearest neighbor regression and classification
- linear regression and classification
- logistic regression and classification
- artificial neural networks

**If we average performance over all imaginable problems, then no method is better than any other. So ANNs must be good for some problems and bad for others.**

**What type of problems are ANNs good for?**

**Any particular strong and weak sides of ANNs?**



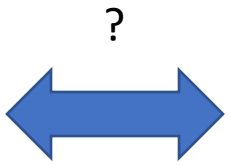
# Is backpropagation inspired by the brain?

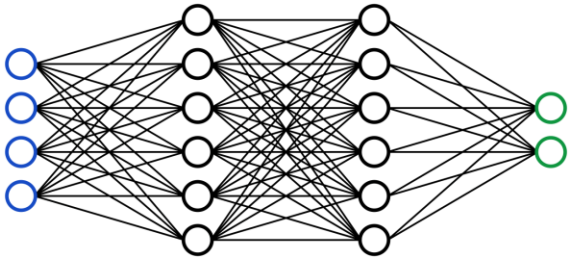
**The structure of perceptrons and ANNs are inspired by the brain.**

**Learning algorithms for networks can also be inspired by the brain (Hebb rule!)**

- **→ What about backpropagation?**
- **→ Any biological link there?**
- **→ And should we even care?**





# Should artificial neural networks be like brain?



# Do artificial neural networks capture the source of intelligence in the brain?

**Modern ANNs abstract away many things that the brain does: Chemical properties, electrical signals, local learning rules**

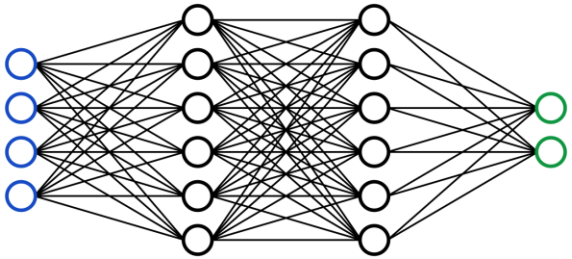
**→ Have ANNs captured the essential parts that make brains intelligent?**

**→ Are chemical, electrical properties NOT important for intelligence? Just a bi-product of evolution?**

**→ Are there things the brain still does much better than ANNs?**



?



# Other learning algorithms for ANNs

**The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).**

**There are many other optimization methods, for example:**

- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

**Are these used, too? Are they better?**

# Other learning algorithms for ANNs

**The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).**

**There are many other optimization methods, for example:**

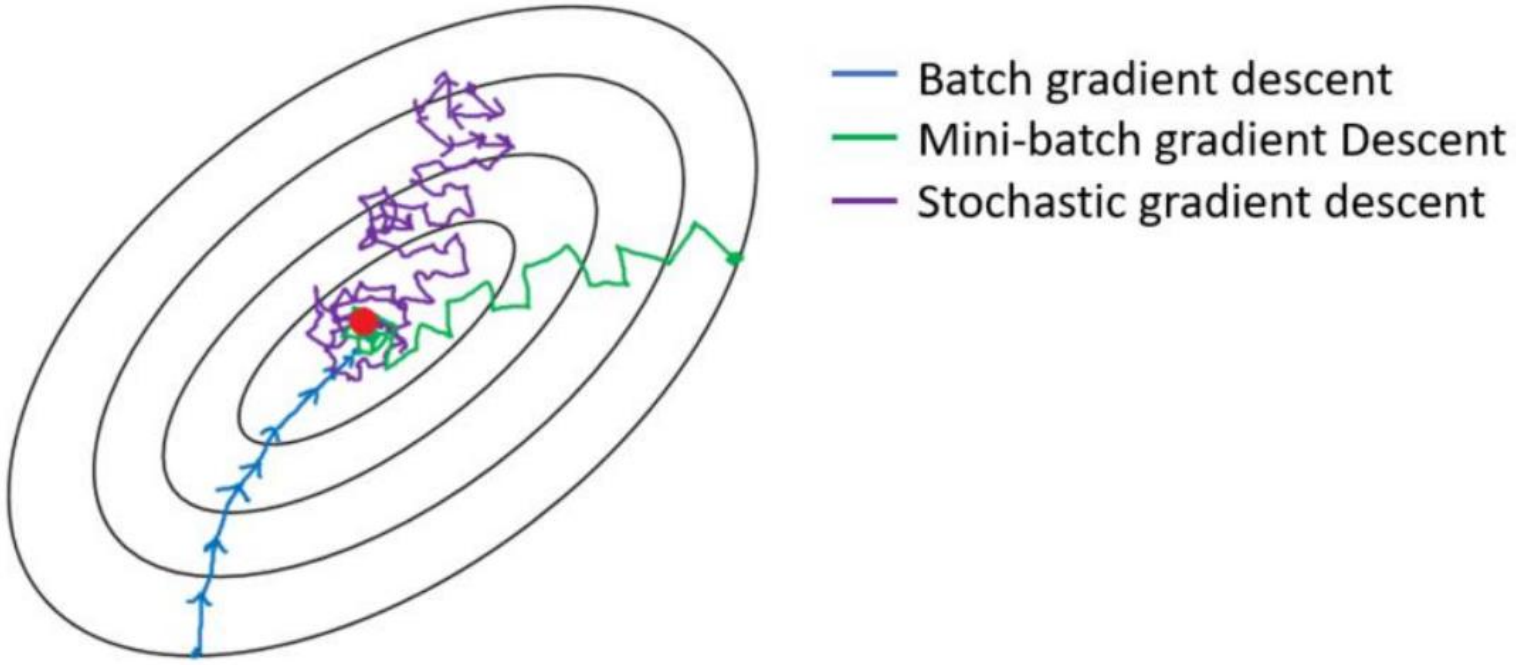
- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

**Are these used, too? Are they better?**

**Short answer: no, almost never (mostly because of computational efficiency)**

# Variants of gradient descent

- Reminder: What is the difference between these? What are their pros/cons?
- Which one is applied most in practice? Why?



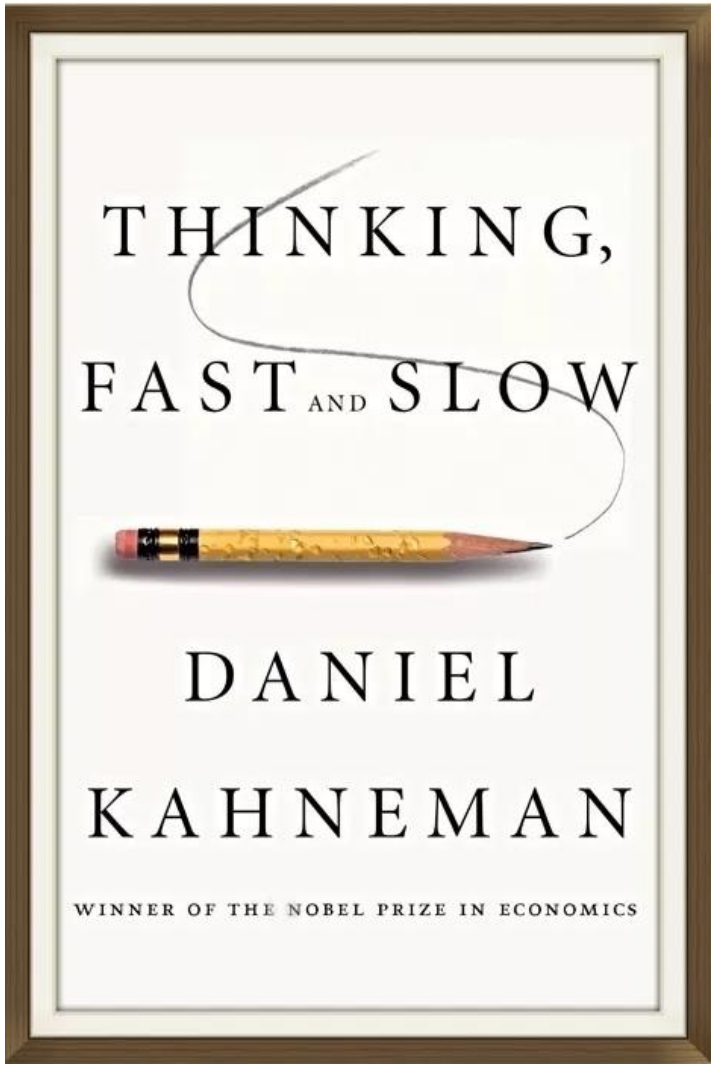
# Fast and slow thinking

It has been hypothesized that humans rely on 2 different modes of thinking:

System 1: Fast, instinctive, emotional

System 2: Slow, deliberative, logical

The best way to make decisions depends on the context



# Fast and slow thinking

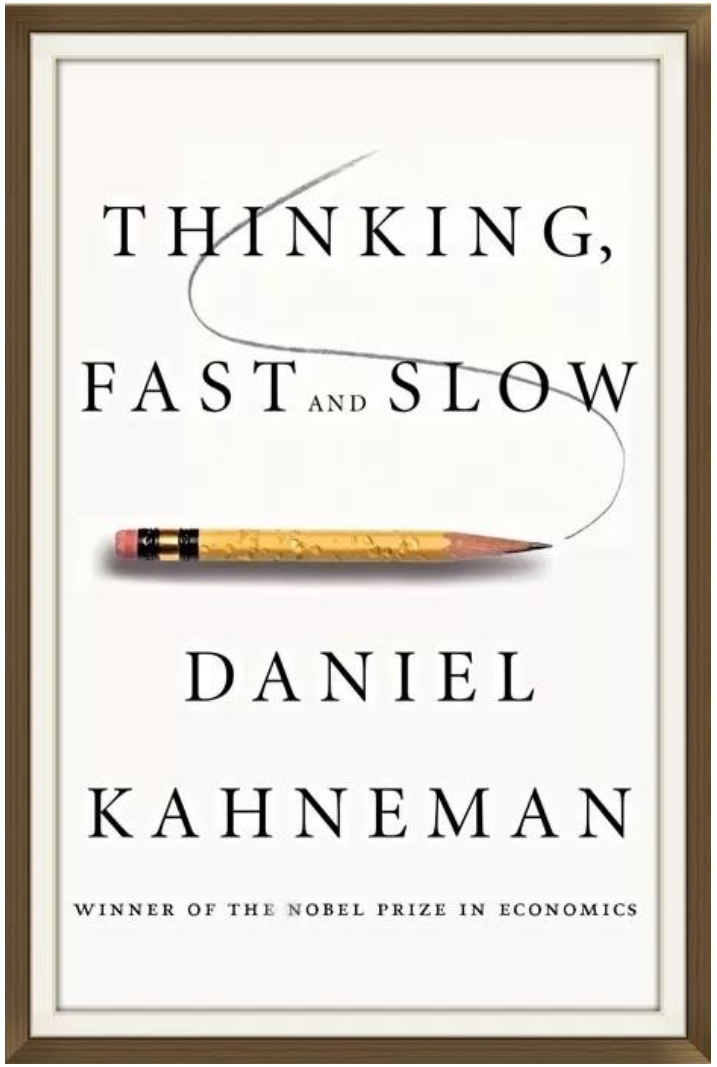
- Some researchers argue that ANNs can only do System 1 (fast) thinking, and that this is limiting:
- First, it makes them less reliable, as they don’t reason deeply before making a decision
- Second, it could mean there are things they can never do, such as long-horizon planning.



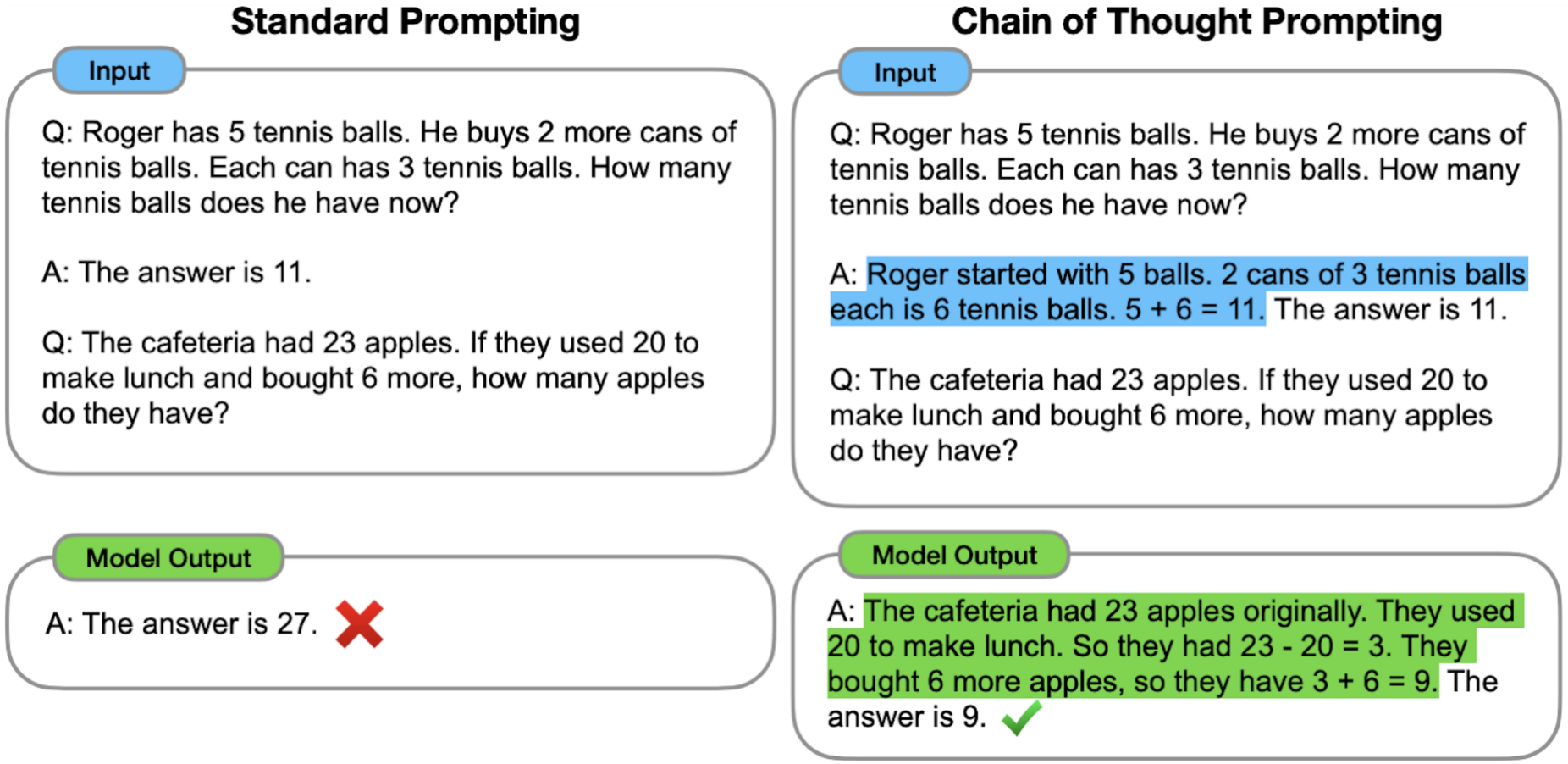

Sample of how an object in a noncanonical orientation and context fools many current object classification systems (Alcorn et al., 2018)

# Fast and slow thinking

- Do you agree neural networks are mainly doing «system 1»-style thinking?
- Do AI agents need to have a «system 2»?
- If so, how could we enable reasoning, planning, logical thinking in future AI agents?



# Fast and slow thinking




We do have the so called "chain-of-thought prompting" in large language models. But it is still very far away from real **reasoning**. *(image: Google Research)*

# One possibility: Neuro-Symbolic AI

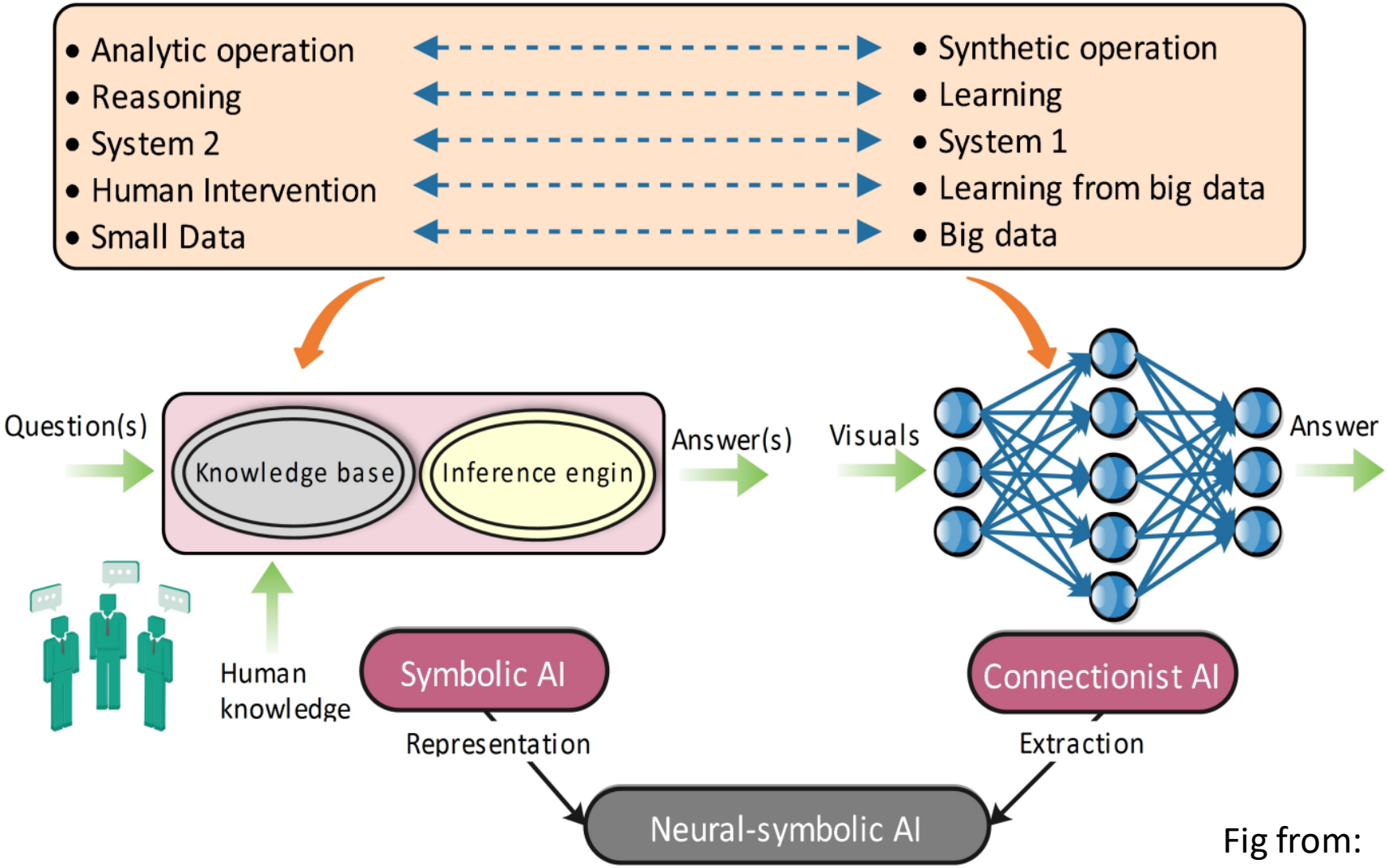



Fig from:
https://arxiv.org/pdf/2208.00374.pdf