01 Introduction, syllabus, MLE vs MAP, why deep learning, foundation models

Syllabus

1. Introduction and foundations
    1. syllabus
    2. why deep learning?
    3. capacity and optimisation
    4. MLE vs MAP
    5. what are foundation models?
    6. computational graph approach, PyTorch
2. Optimisation
    1. modern optimisers and loss functions
    2. generalisation
    3. double descent curve
    4. loss landscape visualisation
    5. Lottery Ticket Hypothesis
    6. Neural Tangent learning
3. CNNs and regularisation
    1. CNNs, ResNet, residual error minimisation approach
    2. evaluation metrics, e.g. perplexity, BLEU, FID
    3. benchmarks and limitations
    4. Regularisation fundamentals: weight decay, early stopping, dropout, batchnorm, layernorm, groupnorm
        1. internal covariate shift
4. Vision Transformer
    1. patches as tokens
    2. embeddings
    3. attention vs convolution inductive bias in CNNs
5. Attention and Transformers
    1. attention as information directing
    2. Transformer architecture details
    3. positional encoding possibilities
    4. layernorm in Transformer
    5. some efficient implementation
6. Sequence models
    1. Limitations of RNNs and LSTMs

    2. state-space models S4 / Mamba
7. Self-supervised learning
    1. contrastive learning approach
    2. masked auto-encoders
    3. possible data augmentation
8. Generative models
    1. VAE, WAE, GAN review
    2. diffusion models, stability
    3. why diffusion is better :?)
9. More advanced diffusion models
10. Scaling laws
    1. what does scaling mean?
    2. few-shot / zero-shot learning
    3. emergent abilities
    4. phase transitions as sudden changes in model spaces
    5. grokking phenomenon
11. Adversarial robustness and interpretability
    1. adversarial examples: why do they exist?
    2. attack methods
    3. connection to interpretability
12. How do Transformers work?
    1. mechanistic interpretation
    2. circuits and induction heads
    3. in-context learning
    4. superposition
    5. feature geometry in Transformer space
13. Practical deployment, future directions
    1. model quantisation and distillation
    2. RLHF - reinforcement learning from human feedback
    3. direct preference optimisation DPO
    4. self correction
    5. multimodality and reasoning
    6. understanding neural networks
    7. artificial general intelligence AGI

Laboratories

1. understanding lectures
2. implementation of models described in lectures

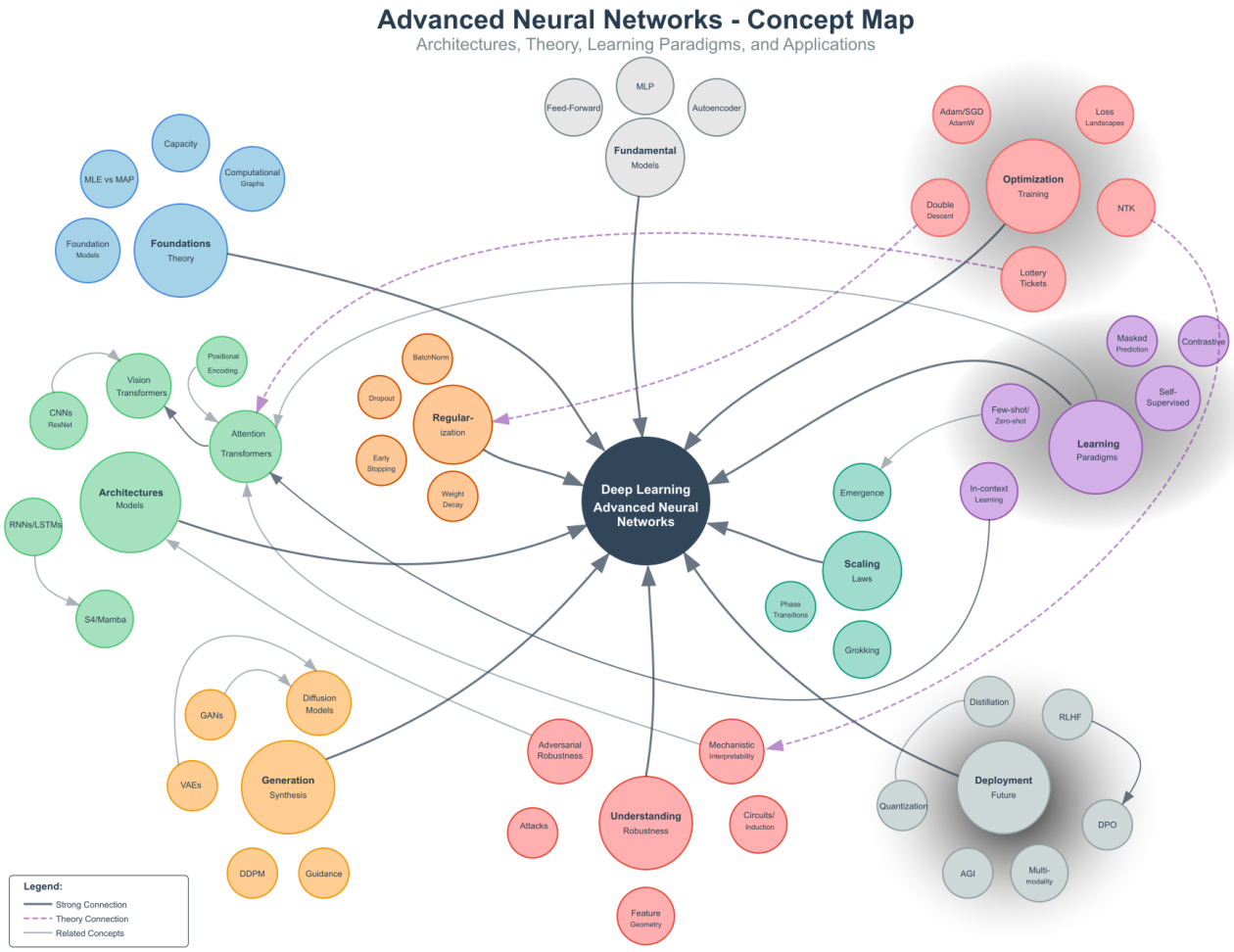
How to pass

1. attention and activity in classes
2. read papers
3. projects
    1. 3-4 in semester connected with current lectures
        1. some projects may be completed in small groups
    2. extra points may be gained by taking part in GMUM projects
        1. projects **have to** be to be extended by implementing some ideas from the lectures
4. passing labs with points above given threshold allow you not to take the exam

Some resources

- PyTorch tutorials
- Dive into Deep Learning https://d2l.ai
- HuggingFace model hub
- Weights and Biases
- etc., etc., etc.

Why deep learning?



Deep Learning Revolution

Traditional ML (before 2012)

- Hand-crafted features (SIFT, HOG, etc.)
- Shallow models (SVM, Random Forests)
- Domain expertise required
- Limited by human insight

Deep Learning (2012-now)

- Automatic feature learning
- Hierarchical representations
- End-to-end optimisation
- Scales with data

---

limitations of shallow learning

Traditional ML tools are limited in processing data in their raw form

- requires careful engineering

- feature design needs considerable domain expertise
  Representation learning
- raw data can be fed
- NN automatically discovers representations
- deep learning maps raw input into consecutive representations slightly more abstract at each step
  - but **all** levels are trained at the same time
  In "standard" understanding of NNs it was thought that

- classifiers can only divide the input raw space into simple regions separated with hyperplanes
- shallow models require feature extractors that "understand" the meaning of data

deep learning

Deep neural networks

- it was suspected that learning useful extractors would be infeasible
- danger that backpropagation learning would get trapped in poor local minima
- practically this rarely is a problem!!!
  - actually the landscape a combinatorial number of saddle points where
  - surface curves up in most dimensions and some in the remainder
  - it does not matter much which saddle points the algorithm gets stuck in
  - there is a very small number of large diameter local minima

---

Representation Learning Hierarchy

```
Pixels   → Edges    → Textures → Parts    → Objects  → Scenes
Layer 1    Layer 2    Layer 3    Layer 4    Layer 5    Layer 6
```

Why "Deep" Works:

- **Compositionality**: Complex concepts from simple ones
  - this is not always the truth
  - NNs predict basing on some strange features, hardly understable, but still the predictions are usually correct
  - still these are compositions of functions
- **Abstraction**: Each layer adds abstraction
- **Reusability**: Lower features shared across tasks
- **Efficiency**: Exponential expressiveness with depth

Mathematical View:

$$f(x) = f_L \quad f_{L-1} \dots f_2 \quad f_1(x)$$

where each layer $f_i(x) = \sigma(W_i x + b_i)$ with nonlinear activation function $\sigma()$

---

success foundation

🖥️ Compute

- GPUs: 1000x speedup (2012)
- TPUs: Another 10x (2016)
- Distributed training at scale

📊 Data

- ImageNet: 1M images (2009)
- Common Crawl: 100TB text
- Self-supervised: Unlimited data

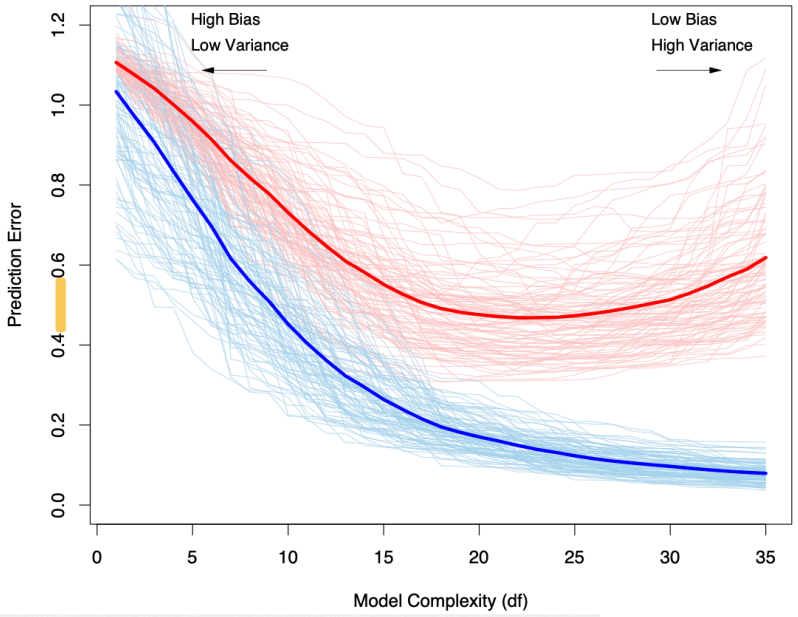
🧱 Algorithms

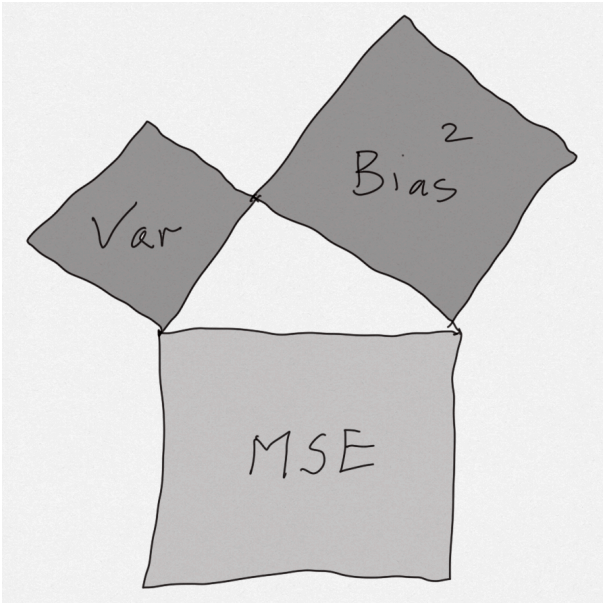
- Backpropagation + autodiff
- Better optimisers (Adam)
- Better architectures (Transformers)

The Perfect Storm: All three arrived together ~2012

Capacity and Optimisation

capacity paradox





Classical ML Wisdom:

```
More parameters → Overfitting → Poor generalization ❌
```

Deep Learning Reality:

```
More parameters → Easier optimization → Better generalization ✅
```

Why This Works:

- Overparameterization creates redundancy
- Multiple paths to good solutions
- Implicit regularization from SGD
- Flat minima generalize better

- it was suspected that learning useful extractors would be infeasible

- for long it the underlying presumption was that backpropagation learning **would get trapped in poor local minima**
- practically this rarely is a problem!!!
    - landscape a combinatorial number of saddle points where
    - surface curves up in most dimensions and some in the remainder
    - it does not matter much which saddle points the algorithm gets stuck in

---

Expressiveness vs Trainability

Expressiveness (What CAN the network represent?)

Universal Approximation Theorem:

- two-layer (one hidden layer with non-linear activations) network can approximate any function
- BUT: May need exponentially many neurons

Depth Advantage:

- Depth-n circuit with polynomial neurons
- Depth-2 circuit needs exponential neurons
- Example: Parity function needs 2^n nodes shallow, n nodes deep

Trainability (Can we FIND good parameters?)

The Optimization Challenge:

- Non-convex: Many local minima
- High-dimensional: Curse of dimensionality
- But empirically: **It works!**

Why Overparameterization Helps

More Parameters = Better Loss Landscape

Narrow Network (few params):

- Isolated minima
- High barriers
- Easy to get stuck

Wide Network (many params):

- Connected minima
- Low barriers
- Many good solutions

Mathematical Intuition:

For loss $L(\theta)$ with n parameters:

- **Critical points**: $L = 0$
- **Probability of local minimum**: $(1/2)^n$
- More parameters → Lower chance of bad minima

---

the blessing of dimensionality

for a long time the very high dimentionality was thought to a **curse**

In High Dimensions:

- Most critical points are saddle points, not minima
- Random directions likely descend
- Volume concentrates near surface

Saddle Point Ratio:

$$\begin{aligned} P(\text{local minimum}) &\approx exp(-\alpha \cdot n) \\ P(\text{saddle point}) &\approx 1 - exp(-\alpha \cdot n) \end{aligned}$$

As $n \to$, almost all critical points are saddles!

- landscape a combinatorial number of saddle points where
- surface curves up in most dimensions and down in the remainder
- it does not matter much which saddle points the algorithm gets stuck in as in most curving down, the value is very close to optimum

Implication is that getting stuck is rare in high dimensions

MLE vs MAP

Maximum Likelihood Estimation (MLE)

The MLE objective seeks to arrive at parameters that maximise the likelihood of observing the data:

$$\theta_{MLE} = \arg\max_{\theta} P(D|\theta) = \arg\max_{\theta} \prod_{i=1}^{N} P(y_i|x_i, \theta)$$

Taking the negative log-likelihood (NLL) to convert product to sum:

$$\theta_{MLE} = \arg\min_{\theta} \left[-\sum_{i=1}^{N} \log P(y_i|x_i, \theta) \right]$$

For regression with Gaussian noise (σ² variance):

$$P(y|x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - f_\theta(x))^2}{2\sigma^2} \right)$$

This gives us the familiar MSE loss:

$$\mathcal{L}_{MLE} = \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2 + \text{const}$$

Maximum A Posteriori (MAP)

MAP includes a prior belief about parameters ($P(\theta|D) = P(D|\theta)P(\theta) \,/\, P(D)$):

$$\theta_{MP} = \arg\max_{\theta} P(\theta|D) = \arg\max_{\theta} P(D|\theta) \cdot P(\theta)$$

Using Bayes' rule and taking negative log:

$$\theta_{MP} = \arg\min_{\theta} \left[-\log P(D|\theta) - \log P(\theta) \right]$$

With Gaussian prior $P(\theta)$ $(0, {}^2 I)$:

$$\log P(\theta) = -\frac{1}{2^2} ||\theta||_2^2 + \text{const}$$

This gives us L2 regularization naturally:

$$\mathcal{L}_{MP} = \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2}_{\text{ata term (ML)}} + \underbrace{\frac{1}{2^2} ||\theta||_2^2}_{\text{Prior term (L2 reg)}}$$

Setting $\lambda = \frac{\sigma^2}{{}^2}$, we get the standard form:

$$\boxed{\mathcal{L}_{MP} = \mathcal{L}_{MLE} + \lambda||\theta||_2^2}$$

Key Insight for Deep Learning

- **MLE**: "What parameters best explain the data?"
- **MAP**: "What parameters best explain the data AND are most probable a priori?"
- **L2 regularization = Gaussian prior assumption**
- **L1 regularization = Laplace prior** (promotes sparsity)

Foundation Models

Foundation Model Paradigm

Old Way (Pre-2018)

```
Task 1 → Train Model 1 from scratch
Task 2 → Train Model 2 from scratch
Task 3 → Train Model 3 from scratch
```

❌ Expensive, data-hungry, no knowledge transfer

- fortunately tasks and models were much smaller then
- the computers were even smaller, so nobody even tried to train them

New Way (Foundation Models)

```
Massive Data → Pretrain ONCE → Foundation Model
                                        ↓
                    Task 1, Task 2, Task 3... Task N
```

✅ Amortised cost, knowledge transfer, few-shot learning

Scale Drives Emergence

The Scaling Laws

Kaplan et al. (2020):

```
Loss ∝ C^(−α) · D^(−β) · P^(−γ)
```

- C = Compute
- D = Data
- P = Parameters

Optimal scaling:

- Parameters: P ∝ C^0.73
- Data: D ∝ C^0.27

Emergent Abilities

Phase Transitions at Scale:

- < 1 B params: Basic pattern matching
- 1-10 B: Language understanding
- 10-100 B: In-context learning
- 100+ B: Reasoning, theory of mind

Foundation Model Zoo

Language

- GPT-4 (>1T params). (1.8 T)
- Claude 3 Opus (2 T)
- Claude 3 Sonnet (175 B params)
- Gemini Ultra (540 B params)
- Claude 4.5 (??? T)

Vision

- CLIP (400M params)
- SAM (630M params)
- DINO v2 (1B params)

Multimodal

- Flamingo (80B params)
- DALL-E 3
- Gemini Vision

Scientific

- AlphaFold (93M params)
- ESMFold (15B params)
- GraphCast (36.7M params)

Key Pattern: One model, many downstream tasks

---

in-context learning

The Revolutionary Capability:

No gradient updates, just prompting:

```
Input: "Translate to French:
        cat → chat
        dog → chien
        bird → ?"
Output: "oiseau"
```

How It Works (Hypothesis):

1. Model implements gradient descent internally
2. Attention = optimization algorithm
3. Context = training data

Mathematical View:

$$f(x, context) \approx f(x) + f\ (\text{context gradient})$$

---

computational graph, PyTorch  , other tools

Some resources used

- "Deep learning", Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Nature 2015
- "Visualising the loss landscape of neural networks", Hao Li et al., arXiv:1712.09913v3
- "Are deep neural networks dramatically overfitted?", Lil'Log, March 2019

- "Double descent demystified ...", Schaeffer et al., arXiv:2303.1415v1
  [Noise, a flaw in human judgement] Daniel Kahneman, Oliver Sibony, Cass R. Sunstein, 2021