Optimization for deep learning

Toulouse School of Economics

Ludovic De Matteis

Course Overview

Planning

	Day	Time	Subject
1	13/10/2025	9:30 - 12:30	Basic definitions, Gradient Descent and Newton's method
2	27/10/2025	9:30 - 12:30	Practical - Gradient Descent and Newton's method
3	03/11/2025	9:30 - 12:30	Neural networks and stockastic gradient descent
4	10/11/2025	9:30 - 12:30	Practical - Neural Networks and digit recognition
5	17/11/2025	9:30 - 12:30	Alternative Neural Structures
6	24/11/2025	9:30 - 12:30	Practical - Alternative Neural structure, Adversial networks

Lecture 3 -

Alternative Neural Structures and practical techniques

Summary

- Convolutional Neural Networks
  - Motivation
  - Convolution and Pooling layers
  - Modern architectures
  - Applications: Image recognition
- Generative Adversarial Networks
  - Motivation
  - Architecture
  - Alternative: diffusion model
- Transformers
  - Motivation
  - Attention is all you need
  - Tranformer blocks
- More techniques in Deep Learning Optimization

Convolutional Neural Networks

Limits of MLP

Input dimensionality explosion

- Images can quickly create large inputs
  - MNIST Digit: $28 * 28 = 784$ pixels
  - CIFAR-10: $32 * 32 * 3 = 3072$ pixels
  - MNIST Digit: $224 * 224 * 3 = 150528$ pixels
- Each pixel is associated to an input neuron
- Large input layers usually require large hidden layers... This create a very large number of parameters
- This makes the network
  - Hard to train
  - Slow to train and to evaluate
  - Prone to overfitting

Limits of MLP

Loss of spatial structure

- The input of MLP needs to be flattened
  - Removes **locality** - adjacent pixels are separated
  - Removes **2D spatial patterns**

No translation invariance

- The network learns to detect an object at a specific place
- If an object moves, the MLP has to relearn new parameters to detect it

We need architectures that exploit local structure and is invariant to rigid transforms => use of convolution

Basics of convolution

Local receptive fields

- Instead of a fully connected input layer
  - A **convolutional filter** looks at a small region of the image
  - The filter is **slided** over the entire image

Convolutional filter / kernels

- A convolutional filter is a small (e.g. $3 * 3$) trainable matrix that aims at detecting specific patterns - edges, corners, shapes, textures...

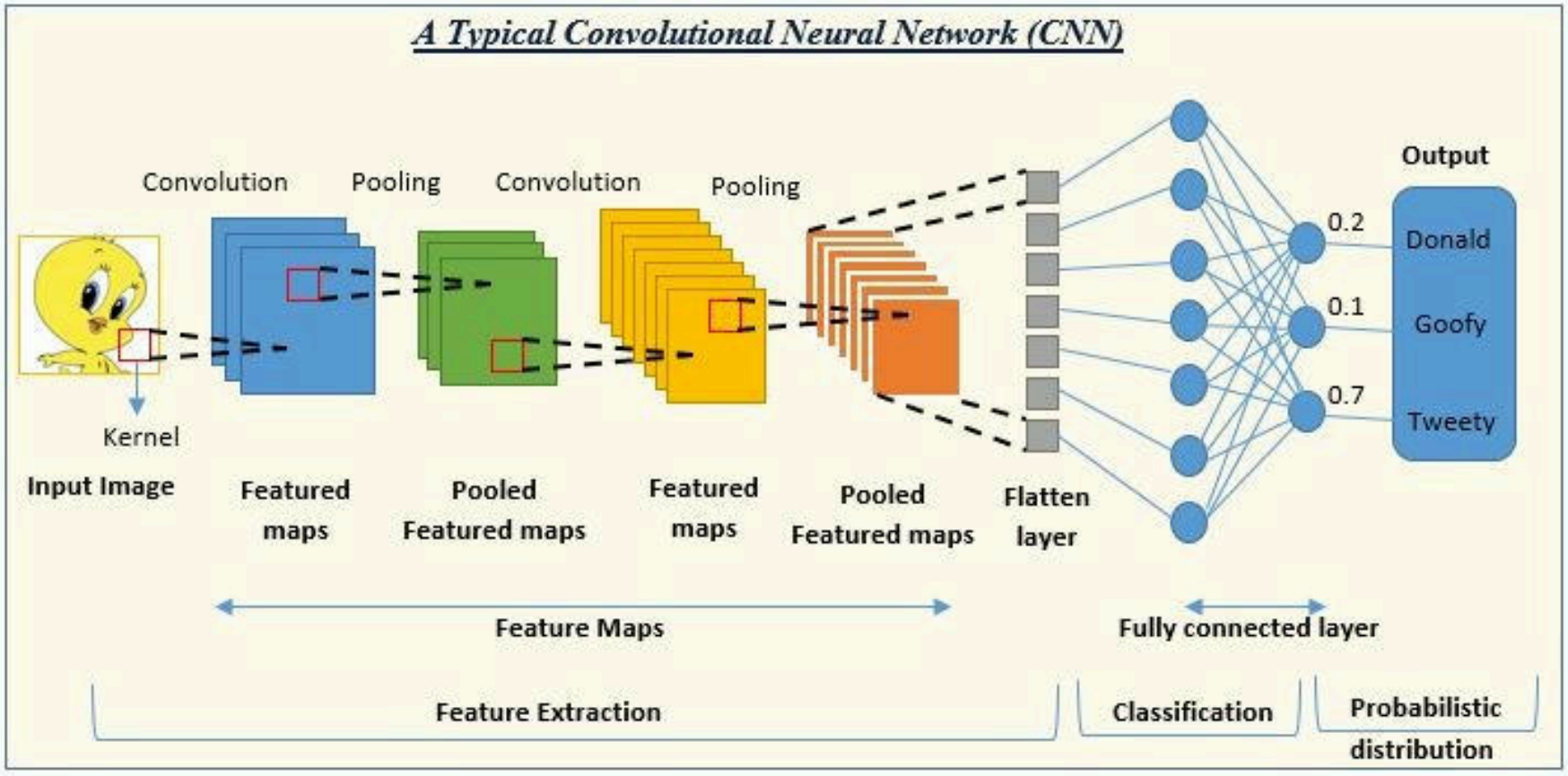
$$\begin{pmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{pmatrix}$$

Basics of convolution

Feature maps

- The same filter is slided across the image, this is called **weight sharing**
- The output of applying a convolutional filter over the entire image is called a **feature map**
- Multiple f ilters can be applied in parallel to extract different features
- The set of feature maps is passed to the next layer
- For 10 filters of size $3 * 3 * 3$, we get 270 parameters in the layer, independently of the input size
- Intuitively, convolutional layers learn to detect local patterns in the input image

CNN Architecture



CNN Layers

Convolution Layer

- The convolution layer is mathematically defined as:

$$Z[i,j,k] = \sum_{u,v,c} X_{i+u,j+v,c} * W_{u,v,c,k} + b_k$$

- Where
  - X is the input tensor of size $H * W * N$
  - Z is the output tensor of size $H' * W' * K$
  - W is the filter tensor of size $F * F * N * K$
  - b is the bias vector of size K

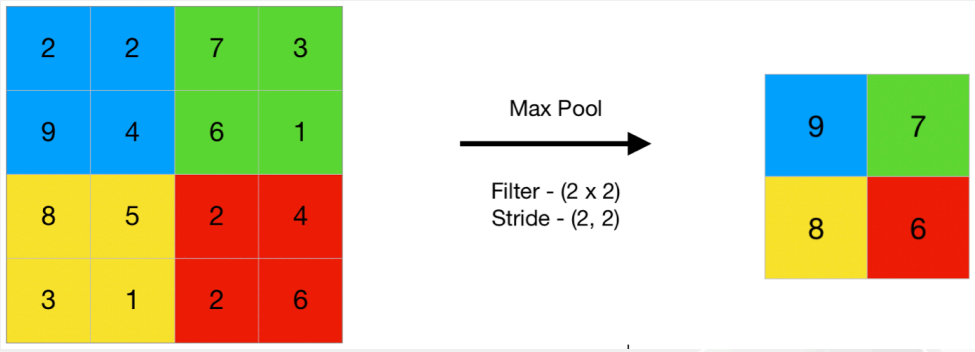
Activation function

- Non-linear activation functions are applied after the convolution operation
  - Most common choice (by far) is ReLU: $f(x) = \max(0, x)$

CNN Layers

Pooling Layer

- Pooling layers are used to reduce the spatial dimensions of the feature maps
- Applies an operation (e.g., max, average) over a local region of the feature map
- Common types of pooling:
  - Max Pooling: takes the maximum value in each region
  - Average Pooling: takes the average value in each region - less common in modern architectures
- Pooling helps to reduce the number of parameters and computation, and also provides some translation invariance



CNN Layers

Fully Connected Layer

- After several convolutional and pooling layers, the feature maps are flattened and passed to one or more fully connected layers
- These layers perform high-level reasoning based on the features extracted by the convolutional layers
- The final layer typically uses a softmax activation function for classification tasks
- Fully connected layers can have a large number of parameters, so they are often kept small in modern CNN architectures

History of modern architectures

- 1989: LeNet-5 (Yann LeCun)
  One of the first successful CNNs for digit recognition
- 2012: AlexNet (Krizhevsky et al.)
  Popularized deep CNNs, won ImageNet competition
- 2014: VGGNet (Simonyan and Zisserman)
  Deeper networks with small $3 * 3$ filters instead of large ones
- 2015: ResNet (He et al.)
  Introduced residual connections to allow training of very deep networks

Practical tricks for CNNs

Data Augmentation

- Techniques to artificially increase the size of the training dataset
  - Random cropping, flipping, rotation, color jittering
- Helps to improve generalization and reduce overfitting

Transfer Learning

- Using pre-trained models on large datasets (e.g., ImageNet) as a starting point for training on a smaller dataset
- Fine-tuning the pre-trained model can lead to better performance with less training data
- Common procedure consists in freezing the early layers and training only the last few layers (classifier)

Practical tricks for CNNs

Regularization Techniques

- Dropout: randomly setting a fraction of the activations to zero during training to prevent co-adaptation of neurons
- Batch Normalization: normalizing the inputs of each layer to stabilize and accelerate training
- Weight Decay: adding an L2 penalty to the loss function to discourage large weights
- Early Stopping: monitoring validation performance and stopping training when performance degrades

Learning Rate Scheduling

- Adjusting the learning rate during training to improve convergence
  - Step decay, exponential decay...

Generative Adversarial Networks

Motivation for GANs

- Generative models aim to learn the underlying distribution of the data to generate new, similar samples
- Traditional generative models (e.g., VAEs) often produce blurry or low-quality images
- GANs provide a framework for training generative models that can produce high-quality, realistic images
- Introduced by Ian Goodfellow in 2014
- Many applications: image synthesis, style transfer, data augmentation…

GAN Architecture

Two main components

- Generator (G): takes random noise as input and generates fake data samples
- Discriminator (D): takes real and fake data samples as input and tries to distinguish between them

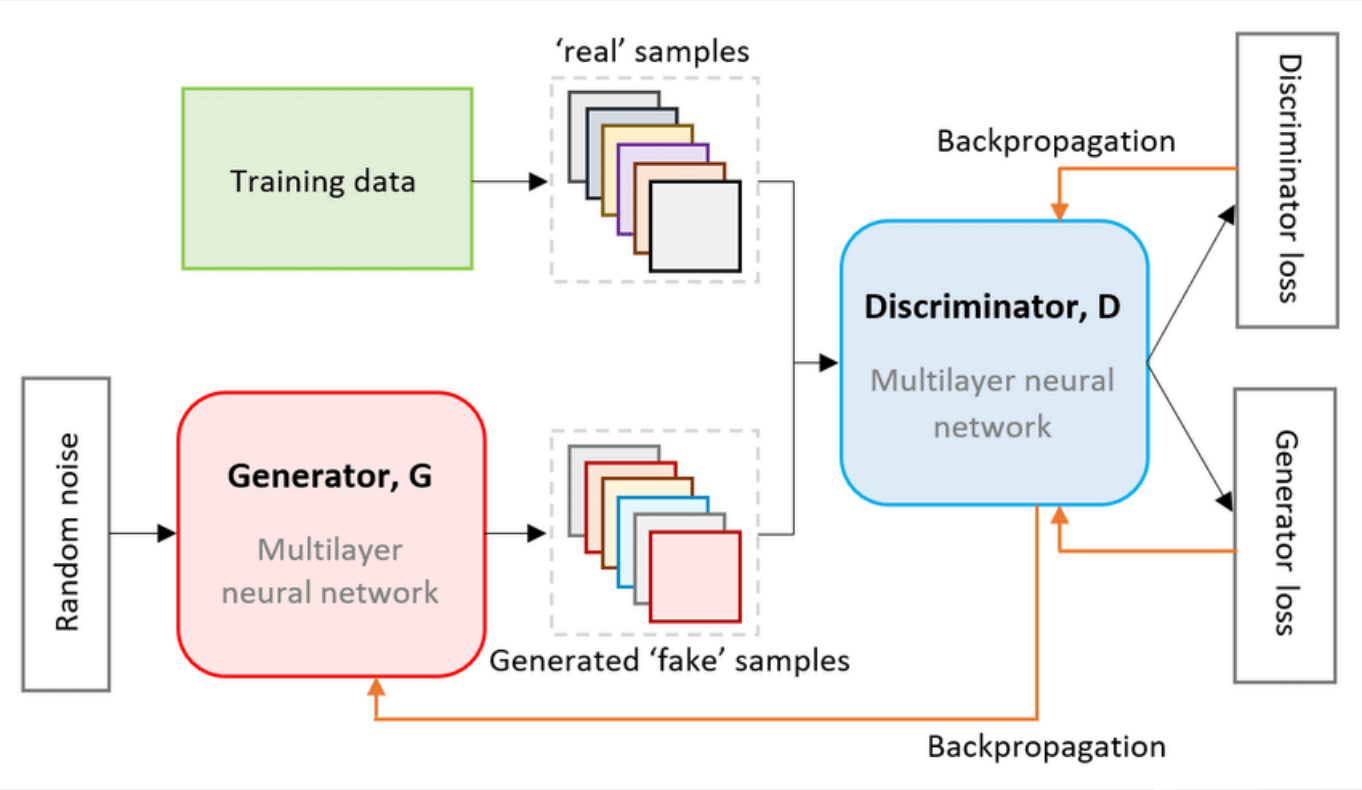
Adversarial training

- The generator and discriminator are trained simultaneously in a minimax game
- The generator aims to produce samples that fool the discriminator, while the discriminator aims to correctly classify real and fake samples
- The objective function is:

$$\min_G \max_D E_{x \sim p_{\text{data}}}[\log D(x)] + E_{z \sim p_z}[\log(1 - D(G(z)))]$$

where p_{data} is the real data distribution and p_z is the noise distribution

GAN Architecture



Training GANs

- Training GANs involves alternating between updating the discriminator and the generator
- Typical training loop:
  1. Sample a batch of real data from the dataset
  2. Sample a batch of noise vectors from the noise distribution
  3. Generate fake data using the generator
  4. Update the discriminator using both real and fake data
  5. Sample another batch of noise vectors
  6. Update the generator to maximize the discriminator's error on the fake data
- Training continues until convergence or for a fixed number of epochs
- GAN training diverges from classical supervised learning as there are two networks competing against each other

Challenges and Improvements

- Training GAN is notoriously difficult due to issues such as:
  - Mode Collapse: the generator produces limited variety of samples
  - Vanishing Gradients: the discriminator becomes too strong, leading to poor generator updates
  - Non-convergence: the training oscillates without reaching a stable point
  - Sensitivity to hyperparameters: learning rate, architecture choices...
- Various techniques have been proposed to improve GAN training:
  - Wasserstein GAN (WGAN): uses a different loss function based on the Wasserstein distance
  - Spectral Normalization: constrains the weights of the discriminator to stabilize training
  - Progressive Growing: gradually increases the resolution of generated images during training

Transformers

Motivation for Transformers

- We want models that can handle sequential data (text, audio, time series)
- Traditional sequence models (RNNs, LSTMs) struggle with long-range dependencies and parallelization
- Transformers introduced by Vaswani et al. in 2017 provide a new architecture based solely on attention mechanisms in the famous paper "Attention is all you need"
- Key advantages:
  - Better handling of long-range dependencies
  - Highly parallelizable, leading to faster training times
  - Scalability to very large models and datasets
- Transformers are the state-of-the-art architecture for many applications
  - Large Language Models (LLMs) like GPT-4, BERT, etc.
  - Vision Transformers (ViT)
  - Reinforcement Learning agents

Attention Mechanism

Scaled Dot-Product Attention

- The core idea of the transformer is the attention mechanism, which allows the model to weigh the importance of different parts of the input sequence when making predictions
- Self-attention computes a representation of the input sequence by relating different positions within the same sequence
- The attention score between two elements is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q (queries), K (keys), and V (values) are linear projections of the input, and d_k is the dimension of the keys

Attention Mechanism

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- The intuition behind this formula is:
  - Compute **similarity scores** between queries and keys using dot product QK^T
  - Scale the scores by the square root of the key dimension to prevent large values
  - Apply softmax to obtain attention **weights** - i.e. a probability distribution
  - Use these weights to compute a weighted sum of the values to create contextualized representations

Multi-head attention

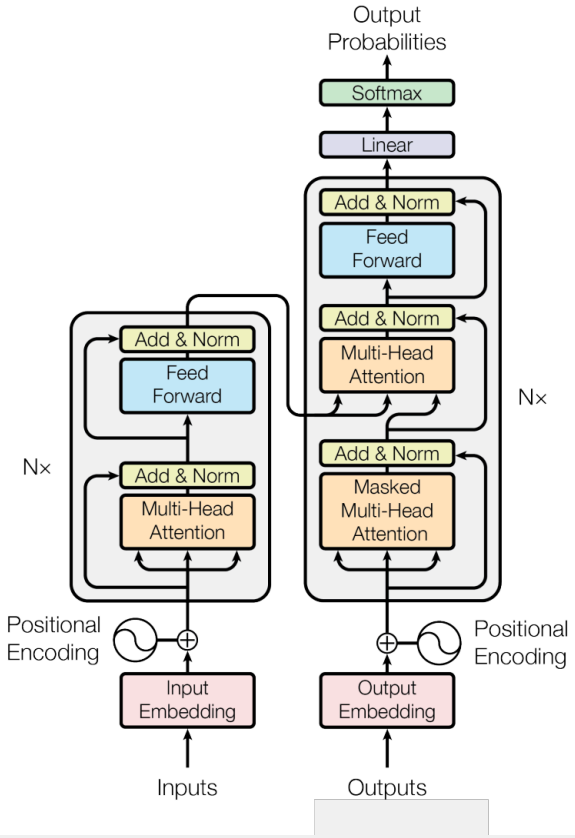
Multi-Head Attention

- Instead of performing a single attention operation, multi-head attention splits the queries, keys, and values into multiple heads
- Each head performs its own attention operation in parallel, allowing the model to capture different aspects of the input - syntactic, semantic, positional...
- The outputs of all heads are concatenated and linearly transformed to produce the final output
- This idea is similar to having multiple filters in CNNs

Tranformers architecture

BERT

Encoder



GPT

Decoder

Transformer Encoder Block

Transformer Encoder Architecture

- A transformer encoder block consists of:
  1. Multi-head self-attention layer
  2. Add & Norm: residual connection followed by layer normalization
  3. Feed-forward neural network (two linear layers with ReLU activation)
  4. Add & Norm: another residual connection followed by layer normalization
- Multiple encoder blocks are stacked to form the full transformer model
- Positional encoding is added to the input embeddings to provide information about the order of the sequence

Transformer Decoder Block

Transformer Decoder Architecture

- A transformer decoder block is similar to the encoder block but includes an additional attention layer:
  1. Masked multi-head self-attention layer (prevents attending to future tokens)
  2. Add & Norm: residual connection followed by layer normalization
  3. Multi-head attention layer over the encoder's output
  4. Add & Norm: another residual connection followed by layer normalization
  5. Feed-forward neural network (two linear layers with ReLU activation)
  6. Add & Norm: final residual connection followed by layer normalization
- The decoder generates the output sequence one token at a time, attending to both the previously generated tokens and the encoder's output

Advantages of Transformers

- Transformers have several advantages over traditional sequence models:
  - Ability to capture long-range dependencies through self-attention
  - Highly parallelizable, leading to faster training times on modern hardware
  - Scalability to very large models and datasets, enabling the training of massive language models
  - Flexibility to handle various types of data, including text, images, and audio
- These advantages have led to transformers becoming the dominant architecture in NLP and other fields

What's more ahead in Optimization in Deep Learning?

- Research is ongoing to improve optimization techniques for training deep learning models
- Some areas of interest include:
  - Adaptive optimization algorithms (e.g., Adam, AdaGrad) that adjust learning rates based on the training dynamics
  - Second-order optimization methods that leverage curvature information for faster convergence
  - Techniques to improve generalization and robustness, such as sharpness-aware minimization (SAM)
  - Exploration of alternative architectures and training paradigms, such as meta-learning and self-supervised learning
- The field is rapidly evolving, with new methods and insights emerging regularly