Advanced Deep Learning in Computer Vision

Dimitrios Papadopoulos
Associate Professor, DTU Compute

Welcome!!!

Dimitrios (Dim) Papadopoulos

- **Associate Professor** at the section of Visual Computing

- **Research interests:** Computer Vision and Machine Learning

- **Prior to DTU:**
  Postdoc, MIT, MA, USA (2018 - 2021)
  Visiting student, ETH Zurich, Switzerland (2016 - 2017)
  PhD, University of Edinburgh, UK (2013 - 2017)
  MSc and Meng at Democritus University of Thrace (2006-2012)

Teachers

Aasa Feragen
Professor at DTU Compute
Section for Visual Computing
afhar@dtu.dk

Dimitrios (Dim) Papadopoulos
Associate Professor at DTU Compute
Section for Visual Computing
dimp@dtu.dk

Teaching Assistants

Onur

monka@dtu.dk

Oskar

ofhkr@dtu.dk

Marco

marscho@dtu.dk

Communication

DTU Learn

https://learn.inside.dtu.dk/d2l/home/296280

- Lectures
- Exercises
- Extra material
- Handins/ Peergrade
- Announcements

Slack workspace

https://join.slack.com/t/02501adlcv2026/shared_invite/zt-3p53x6mtg-dDtiUZwVbih5_DZJI5AL_Q

- Group members
- Questions for exercises
- General questions
- Announcements

Computation (GPUs??)

- **No dedicated nodes**

- **Exercises:**
  (a) Colab
  (b) local CPU/GPU
  (c) HPC batch jobs
  (d) HPC interactive

- **Projects:**
  HPC batch jobs

Intro Deep Learning in Computer Vision (January)

IMAGE CLASSIFICATION

cat

OBJECT DETECTION

IMAGE SEGMENTATION

Advanced Deep Learning in Computer Vision

Advanced Deep Learning in Computer Vision

IMAGE CLASSIFICATION

cat

OBJECT DETECTION

IMAGE SEGMENTATION

IMAGE GENERATION

Advanced Deep Learning in Computer Vision

IMAGE CLASSIFICATION

OBJECT DETECTION

State-of-the-art Visual Recognition: Beyond CNNs

State-of-the-art Image Generation

Sequence (Language) Modeling

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Vision and Language

“a cute cat in Copenhagen”

What is the mustache made of?

Beyond Still Images: Video Understanding

Transformers

Diffusion models

Join at menti.com | use code 5494 8990

Name 3 AI tools that you know, use, etc…

Rules:

- No capital letters
- One word only, concat words if multiple
- e.g. chatgpt, claudecode, googlegemini

AI Tools & Models

The 2024-2025 Landscape

Chat/LLM · Image Gen · Video · Code · Search · Audio

Practicalities

Evaluation

Exams

Schedule

Tue 3/2	13.00-15.00 Lecture Introduction to the course, Transformer, Attention, and Encoders 15.00-17.00 Exercise 1.1	Dim TAs
Tue 10/2	13.00-15.00 Lecture Visual Transformers and applications to segmentation, and detection 15.00-17.00 Exercise 1.2	Dim TAs
Tue 17/2	13.00-15.00 Lecture Transformer Decoder, Large Language Models (LLM), and GPTs 15.00-17.00 Exercise 1.3	Dim TAs
Mon 23/2	Hand-in Assignment 1	
Tue 24/2	13.00-15.00 Lecture Generative models: DDPM 15.00-17.00 Exercise 2.1	Aasa TAs
Tue 3/3	13.00-15.00 Lecture Generative models: Diffusion Guidance 15.00-17.00 Exercise 2.2	Aasa TAs
Tue 10/3	13.00-15.00 Lecture Generative models: Latent Diffusion Model, CLIP and Text2Image generation 15.00-17.00 Exercise 2.3	Dim TAs
Mon 16/3	Hand-in Assignment 2	
Tue 17/3	13.00-16.00 Reading Group 16.00-17.00 Project Pitch	All
Tue 24/3	13.00-15.00 Lecture Self-supervised Learning and Training with limited labels 15.00-17.00 Project work	Dim TAs
Tue 31/3	Easter holiday	
Tue 7/4	13.00-15.00 Lecture Multimodal learning and Vision Language Models (VLMs) 15.00-17.00 Project work	Dim TAs
Tue 14/4	13.00-15.00 Lecture TBA 15.00-17.00 Project work	Stella TAs
Tue 21/4	13.00-15.00 Lecture Explainability 15.00-17.00 Project work	Aasa TAs
Tue 28/4	13.00-15.00 Lecture Fairness and Bias and Ethics 15.00-17.00 Supervision meetings (Final)	Aasa All
Fri 1/5	Project Hand-in**	
Tue 5/5	13.00- Project session and final feedback	All
Fri 15/5**	Final project Hand-in	
Tue 26/5**	Oral exams	

Schedule

Tue 3/2	13.00-15.00 Lecture Introduction to the course, Transformer, Attention, and Encoders 15.00-17.00 Exercise 1.1	Dim TAs
Tue 10/2	13.00-15.00 Lecture Visual Transformers and applications to segmentation, and detection 15.00-17.00 Exercise 1.2	Dim TAs
Tue 17/2	13.00-15.00 Lecture Transformer Decoder, Large Language Models (LLM), and GPTs 15.00-17.00 Exercise 1.3	Dim TAs
Mon 23/2	Hand-in Assignment 1	
Tue 24/2	13.00-15.00 Lecture Generative models: DDPM 15.00-17.00 Exercise 2.1	Aasa TAs
Tue 3/3	13.00-15.00 Lecture Generative models: Diffusion Guidance 15.00-17.00 Exercise 2.2	Aasa TAs
Tue 10/3	13.00-15.00 Lecture Generative models: Latent Diffusion Model, CLIP and Text2Image generation 15.00-17.00 Exercise 2.3	Dim TAs
Mon 16/3	Hand-in Assignment 2	
Tue 17/3	13.00-16.00 Reading Group 16.00-17.00 Project Pitch	All
Tue 24/3	13.00-15.00 Lecture Self-supervised Learning and Training with limited labels 15.00-17.00 Project work	Dim TAs
Tue 31/3	Easter holiday	
Tue 7/4	13.00-15.00 Lecture Multimodal learning and Vision Language Models (VLMs) 15.00-17.00 Project work	Dim TAs
Tue 14/4	13.00-15.00 Lecture TBA 15.00-17.00 Project work	Stella TAs
Tue 21/4	13.00-15.00 Lecture Explainability 15.00-17.00 Project work	Aasa TAs
Tue 28/4	13.00-15.00 Lecture Fairness and Bias and Ethics 15.00-17.00 Supervision meetings (Final)	Aasa All
Fri 1/5	Project Hand-in**	
Tue 5/5	13.00- Project session and final feedback	All
Fri 15/5**	Final project Hand-in	
Tue 26/5**	Oral exams	

- **Lectures:** Every Tuesday at 13:00
- **Room:** Building 306, aud 32
- **Group work:** B306: H097, H096, H108b, H108a
- **TA hours:** 15.00-17.00

Schedule

Tue 3/2	13.00-15.00 Lecture Introduction to the course, Transformer, Attention, and Encoders 15.00-17.00 Exercise 1.1	Dim TAs
Tue 10/2	13.00-15.00 Lecture Visual Transformers and applications to segmentation, and detection 15.00-17.00 Exercise 1.2	Dim TAs
Tue 17/2	13.00-15.00 Lecture Transformer Decoder, Large Language Models (LLM), and GPTs 15.00-17.00 Exercise 1.3	Dim TAs
Mon 23/2	Hand-in Assignment 1	
Tue 24/2	13.00-15.00 Lecture Generative models: DDPM 15.00-17.00 Exercise 2.1	Aasa TAs
Tue 3/3	13.00-15.00 Lecture Generative models: Diffusion Guidance 15.00-17.00 Exercise 2.2	Aasa TAs
Tue 10/3	13.00-15.00 Lecture Generative models: Latent Diffusion Model, CLIP and Text2Image generation 15.00-17.00 Exercise 2.3	Dim TAs
Mon 16/3	Hand-in Assignment 2	
Tue 17/3	13.00-16.00 Reading Group 16.00-17.00 Project Pitch	All
Tue 24/3	13.00-15.00 Lecture Self-supervised Learning and Training with limited labels 15.00-17.00 Project work	Dim TAs
Tue 31/3	Easter holiday	
Tue 7/4	13.00-15.00 Lecture Multimodal learning and Vision Language Models (VLMs) 15.00-17.00 Project work	Dim TAs
Tue 14/4	13.00-15.00 Lecture TBA 15.00-17.00 Project work	Stella TAs
Tue 21/4	13.00-15.00 Lecture Explainability 15.00-17.00 Project work	Aasa TAs
Tue 28/4	13.00-15.00 Lecture Fairness and Bias and Ethics 15.00-17.00 Supervision meetings (Final)	Aasa All
Fri 1/5	Project Hand-in**	
Tue 5/5	13.00- Project session and final feedback	All
Fri 15/5**	Final project Hand-in	
Tue 26/5**	Oral exams	

- **Lectures:** Every Tuesday at 13:00
- **Room:** Building 306, aud 32
- **Group work:** B306: H097, H096, H108b, H108a
- **TA hours:** 15.00-17.00

- Fully physical course
- 5 ECTS (~9 hours per week), 13 weeks
- No textbook
- Lectures, Exercises, Paper, Lecture notes, References
- https://visionbook.mit.edu/

Schedule

Tue 3/2	13.00-15.00 Lecture Introduction to the course, Transformer, Attention, and Encoders 15.00-17.00 Exercise 1.1	Dim TAs
Tue 10/2	13.00-15.00 Lecture Visual Transformers and applications to segmentation, and detection 15.00-17.00 Exercise 1.2	Dim TAs
Tue 17/2	13.00-15.00 Lecture Transformer Decoder, Large Language Models (LLM), and GPTs 15.00-17.00 Exercise 1.3	Dim TAs
Mon 23/2	Hand-in Assignment 1	
Tue 24/2	13.00-15.00 Lecture Generative models: DDPM 15.00-17.00 Exercise 2.1	Aasa TAs
Tue 3/3	13.00-15.00 Lecture Generative models: Diffusion Guidance 15.00-17.00 Exercise 2.2	Aasa TAs
Tue 10/3	13.00-15.00 Lecture Generative models: Latent Diffusion Model, CLIP and Text2Image generation 15.00-17.00 Exercise 2.3	Dim TAs
Mon 16/3	Hand-in Assignment 2	
Tue 17/3	13.00-16.00 Reading Group 16.00-17.00 Project Pitch	All
Tue 24/3	13.00-15.00 Lecture Self-supervised Learning and Training with limited labels 15.00-17.00 Project work	Dim TAs
Tue 31/3	Easter holiday	
Tue 7/4	13.00-15.00 Lecture Multimodal learning and Vision Language Models (VLMs) 15.00-17.00 Project work	Dim TAs
Tue 14/4	13.00-15.00 Lecture TBA 15.00-17.00 Project work	Stella TAs
Tue 21/4	13.00-15.00 Lecture Explainability 15.00-17.00 Project work	Aasa TAs
Tue 28/4	13.00-15.00 Lecture Fairness and Bias and Ethics 15.00-17.00 Supervision meetings (Final)	Aasa All
Fri 1/5	Project Hand-in**	
Tue 5/5	13.00- Project session and final feedback	All
Fri 15/5**	Final project Hand-in	
Tue 26/5**	Oral exams	

- **Lectures:** Every Tuesday at 13:00
- **Room:** Building 306, aud 32
- **Group work:** B306: H097, H096, H108b, H108a
- **TA hours:** 15.00-17.00

- Fully physical course
- 5 ECTS (~9 hours per week), 13 weeks
- No textbook
- Lectures, Exercises, Paper, Lecture notes, References
- https://visionbook.mit.edu/

- **Weekly exercises and Assignments**
- **Final project**
- **Lectures every week**

Weekly exercises and Assignments

Tue 3/2	13.00-15.00 Lecture Introduction to the course, Transformer, Attention, and Encoders 15.00-17.00 Exercise 1.1
Tue 10/2	13.00-15.00 Lecture Visual Transformers and applications to segmentation, and detection 15.00-17.00 Exercise 1.2
Tue 17/2	13.00-15.00 Lecture Transformer Decoder, Large Language Models (LLM), and GPTs 15.00-17.00 Exercise 1.3
Mon 23/2	Hand-in Assignment 1
Tue 24/2	13.00-15.00 Lecture Generative models: DDPM 15.00-17.00 Exercise 2.1
Tue 3/3	13.00-15.00 Lecture Generative models: Diffusion Guidance 15.00-17.00 Exercise 2.2
Tue 10/3	13.00-15.00 Lecture Generative models: Latent Diffusion Model, CLIP and Text2Image generation 15.00-17.00 Exercise 2.3
Mon 16/3	Hand-in Assignment 2

- **6** first weeks, **6** weekly exercises
- **Individual** and **Groups (3-4 students)**
- **2** deliverables/ reports in groups
- Up to **4** pages. Recommendation: use **template**
- Peergrading and TA feedback
- **Pass/Fail**

Final Project

Tue 17/3	13.00-16.00 Reading Group 16.00-17.00 Project Pitch
Tue 24/3	13.00-15.00 Lecture Self-supervised Learning and Training with limited labels 15.00-17.00 Project work
Tue 31/3	Easter holiday
Tue 7/4	13.00-15.00 Lecture Multimodal learning and Vision Language Models (VLMs) 15.00-17.00 Project work
Tue 14/4	13.00-15.00 Lecture TBA 15.00-17.00 Project work
Tue 21/4	13.00-15.00 Lecture Explainability 15.00-17.00 Project work
Tue 28/4	13.00-15.00 Lecture Fairness and Bias and Ethics 15.00-17.00 Supervision meetings (Final)
Fri 1/5	Project Hand-in**
Tue 5/5	13.00- Project session and final feedback
Fri 15/5**	Final project Hand-in
Tue 26/5**	Oral exams

- 7 weeks + easter week
- **Groups up to 4**
- Deliverables: poster
- Select one from the proposed proposals or propose your own
- 1-on-1 supervision meetings (Start-Mid-Final)
- Poster session and final feedback
- **Oral Exams:** project + general questions

Practicalities

Lectures: Every Tuesday at 13:00, Building 306, aud 32

- **Weekly exercises** and **Peergrading (Individual and Groups)**
  - First 6 weeks
  - 2 Deliverables in total (group)
  - Pass/Fail
  - Peergrading + TA feedback

- **Final project (Groups up to 4)**
  - Start: 17 March
  - Poster session: 5 May
  - Hand-in: 15 May
  - Deliverable: **Poster**
  - Exam: 26** May - **Oral** (**project** + **general questions**)

Prerequisites

- **02450 Intro to Machine Learning**
- **02456 Deep Learning**
- **02516 Introduction to Deep Learning in Computer Vision**

(or similar)

Prerequisites

- **02450 Intro to Machine Learning**
- **02456 Deep Learning**
- **02516 Introduction to Deep Learning in Computer Vision**

(or similar)

Experience with
- **Building** and **training** deep neural nets
- For **Visual recognition tasks** (classification, detection, segmentation)
- On **custom** data
- Using **PyTorch**
- On a remote server **GPU**

Today

- **Lecture: 13:00 – 15:00**

- **Exercise: 15:00 – 17:00 (with TAs)**

Today

- **Lecture: 13:00 – 15:00**

- **Exercise: 15:00 – 17:00 (with TAs)**

Today

- **Lecture: 13:00 – 15:00**

- **Exercise: 15:00 – 17:00 (with TAs)**

Sequence to sequence Models

Sequence to sequence Models

Sequence to sequence Model

Sequence to sequence Models

This course is amazing

Sequence to sequence Model

Sequence to sequence Models

This course is amazing

Sequence to sequence Model

Dette kursus er fantastisk

Neural Nets and CNNs

Neural Nets and CNNs

- ConvNets and vanilla neural networks use a fixed-sized input and ouput
- Some tasks require a sequence of input and/or output vectors
- Limited receptive field

Neural Nets and CNNs

Neural Nets and CNNs

Partial sequence of both input and output

CL] 14 Dec 2014

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence

Recurrent Neural Networks (RNNs)

Backpropagation Through Time: What It Does and How to Do It

PAUL J. WERBOS

Backpropagation is now the most widely used tool in the field of artificial neural networks. At the core of backpropagation is a method for calculating derivatives exactly and efficiently in any large system made up of elementary subsystems or calculations which are represented by known, differentiable functions; thus, backpropagation has many applications which do not involve neural networks as such.

This paper first reviews basic backpropagation, a simple method which is now being widely used in areas like pattern recognition and fault diagnosis. Next, it presents the basic equations for backpropagation through time, and discusses applications to areas like pattern recognition involving dynamic systems, systems identification, and control. Finally, it describes further extensions of this method, to deal with systems other than neural networks, systems involving simultaneous equations or true recurrent networks, and other practical issues which arise with this method. Pseudocode is provided to clarify the algorithms. The chain rule for ordered derivatives—the theorem which underlies backpropagation—is briefly discussed.

I. Introduction

Backpropagation through time is a very powerful tool, with applications to pattern recognition, dynamic modeling, sensitivity analysis, and the control of systems over time, among others. It can be applied to neural networks, to econometric models, to fuzzy logic structures, to fluid dynamics models, and to almost any system built up from elementary subsystems or calculations. The one serious constraint is that the elementary subsystems must be represented by functions known to the user, functions which are both continuous and differentiable (i.e., possess derivatives). For example, the first practical application of backpropagation was for estimating a dynamic model to predict nationalism and social communications in 1974 [1].

Unfortunately, the most general formulation of backpropagation can only be used by those who are willing to work out the mathematics of their particular application. This paper will mainly describe a simpler version of backpropagation, which can be translated into computer code and applied directly by neural network users.

Section II will review the simplest and most widely used form of backpropagation, which may be called "basic back-

Manuscript received September 12, 1989; revised March 15, 1990.
The author is with the National Science Foundation, 1800 G St. NW, Washington, DC 20550.
IEEE Log Number 9039172.

U.S. Government work not protected by U.S. copyright

propagation." The concepts here will
those who have read the paper by Rum
Williams [2] in the seminal book *Par*
cessing, which played a pivotal role in
the field. (That book also acknowledg
Parker [3] and Le Cun [4], and the pi
Smith of the Systems Development Fo
tion will use new notation which adds a
makes it easier to go on to complex
orous manner. (The need for new
unnecessary to some, but for those wh
propagation to complex systems, it is

Section III will use the same notati
propagation through time. Backpropa
has been applied to concrete proble
authors, including, at least, Watrous a
and Waibel *et al.* [6], Nguyen and W
Kawato [9], Elman and Zipser, Narend
[11], [12], [15]. Section IV will discuss w
simplified discussion, and how to do

At its core, backpropagation is sim
exact method for calculating all the d
target quantity (such as pattern class
respect to a large set of input quantitie
eters or weights in a classification ru
through time extends this method
dynamic systems. This allows one to
tives needed when optimizing an it
cedure, a neural network with memor
which maximizes performance over t

II. Basic Backpropagation

A. The Supervised Learning Problem

Basic backpropagation is current
method for performing the supervised
is symbolized in Fig. 1.

In supervised learning, we try to ada
network so that its actual outputs (Y)
target outputs (Y) for a training set w
terns. The goal is to adapt the param
so that it performs well for patterns fr
ing set.

The main use of supervised learning

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on August 29,2023 at 01:05:05 UTC from IEEE Xplore. Restrictions apply.

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \quad (2)$$

† To whom correspondence should be addressed.

©1986 Nature Publishing Group

LONG SHORT-TERM MEMORY

Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
echnische Universität München
80290 München, Germany
hreit@informatik.tu-muenchen.de
/www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Abstract

ning to store information over extended time intervals via recurrent backpropagation
very long time, mostly due to insufficient, decaying error back flow. We briefly review
ter's 1991 analysis of this problem, then address it by introducing a novel, efficient,
-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient
his does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000
time steps by enforcing *constant* error flow through "constant error carrousels" within
units. Multiplicative gate units learn to open and close access to the constant error
STM is local in space and time; its computational complexity per time step and weight
Our experiments with artificial data involve local, distributed, real-valued, and noisy
representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation,
nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and
uch faster. LSTM also solves complex, artificial long time lag tasks that have never
ved by previous recurrent network algorithms.

RODUCTION

etworks can in principle use their feedback connections to store representations of
events in form of activations ("short-term memory", as opposed to "long-term mem-
ed by slowly changing weights). This is potentially significant for many applications,
eech processing, non-Markovian control, and music composition (e.g., Mozer 1992).
dely used algorithms for learning *what* to put in short-term memory, however, take
e or do not work well at all, especially when minimal time lags between inputs and
g teacher signals are long. Although theoretically fascinating, existing methods do
clear *practical* advantages over, say, backprop in feedforward nets with limited time
is paper will review an analysis of the problem and suggest a remedy.

blem. With conventional "Back-Propagation Through Time" (BPTT, e.g., Williams
992, Werbos 1988) or "Real-Time Recurrent Learning" (RTRL, e.g., Robinson and
), error signals "flowing backwards in time" tend to either (1) blow up or (2) vanish:
l evolution of the backpropagated error exponentially depends on the size of the
hreiter 1991). Case (1) may lead to oscillating weights, while in case (2) learning to
ime lags takes a prohibitive amount of time, or does not work at all (see section 3).

ıedy. This paper presents *"Long Short-Term Memory"* (LSTM), a novel recurrent
itecture in conjunction with an appropriate gradient-based learning algorithm. LSTM
o overcome these error back-flow problems. It can learn to bridge time intervals in
0 steps even in case of noisy, incompressible input sequences, without loss of short
abilities. This is achieved by an efficient, gradient-based algorithm for an architecture

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs)

$$h_t = g_h(W_h[h_{t-1}; x_t] + b_h) \qquad y_t = g_y(W_y h_t + b_y)$$

How do we train an RNN?

$h_t = g_h(W_h[h_{t-1}; x_t] + b_h)$ $\quad$ $y_t = g_y(W_y h_t + b_y)$

How do we train an RNN?

Backpropagation Through Time: What It Does and How to Do It

PAUL J. WERBOS

Backpropagation is now the most widely used tool in the field of artificial neural networks. At the core of backpropagation is a method for calculating derivatives exactly and efficiently in any large system made up of elementary subsystems or calculations which are represented by known, differentiable functions; thus, backpropagation has many applications which do not involve neural networks as such.

This paper first reviews basic backpropagation, a simple method which is now being widely used in areas like pattern recognition and fault diagnosis. Next, it presents the basic equations for backpropagation through time, and discusses applications to areas like pattern recognition involving dynamic systems, systems identification, and control. Finally, it describes further extensions of this method, to deal with systems other than neural networks, systems involving simultaneous equations or true recurrent networks, and other practical issues which arise with this method. Pseudocode is provided to clarify the algorithms. The chain rule for ordered derivatives—the theorem which underlies backpropagation—is briefly discussed.

propagation." The concepts here will already be familiar to those who have read the paper by Rumelhart, Hinton, and Williams [2] in the seminal book *Parallel Distributed Processing*, which played a pivotal role in the development of the field. (That book also acknowledged the prior work of Parker [3] and Le Cun [4], and the pivotal role of Charles Smith of the Systems Development Foundation.) This section will use new notation which adds a bit of generality and makes it easier to go on to complex applications in a rigorous manner. (The need for new notation may seem unnecessary to some, but for those who have to *apply* backpropagation to complex systems, it is essential.)

Section III will use the same notation to describe backpropagation through time. Backpropagation through time has been applied to concrete problems by a number of authors, including, at least, Watrous and Shastri [5], Sawai

$h_t = g_h(W_h[h_{t-1}; x_t] + b_h)$ $\quad$ $y_t = g_y(W_y h_t + b_y)$

- ***Backward to compute gradients***
- ***Backprop consists of summing over all paths from losses to W***

$$\frac{\partial L}{\partial \mathrm{vec}(\mathbf{W})^\top} = \sum_{n=0}^{N} \sum_{k=0}^{n} \frac{\partial L_n}{\partial y_n} \frac{\partial y_n}{\partial h_n^\top} \frac{\partial h_n}{\partial h_k^\top} \frac{\partial h_k}{\partial \mathrm{vec}(\mathbf{W})^\top}$$

$$\frac{\partial h_K}{\partial h_0^\top} = \frac{\partial h_K}{\partial h_{K-1}^\top} \frac{\partial h_{K-1}}{\partial h_{K-2}^\top} \cdots \frac{\partial h_1}{\partial h_0^\top}$$

What's wrong with RNNs?

$$\frac{\partial L}{\partial \mathrm{vec}(\mathbf{W})^\top} = \sum_{n=0}^{N} \sum_{k=0}^{n} \frac{\partial L_n}{\partial y_n} \frac{\partial y_n}{\partial h_n^\top} \frac{\partial h_n}{\partial h_k^\top} \frac{\partial h_k}{\partial \mathrm{vec}(\mathbf{W})^\top}$$

$$\frac{\partial h_K}{\partial h_0^\top} = \frac{\partial h_K}{\partial h_{K-1}^\top} \frac{\partial h_{K-1}}{\partial h_{K-2}^\top} \cdots \frac{\partial h_1}{\partial h_0^\top}$$

$h_t = g_h(W_h[h_{t-1}; x_t] + b_h)$ $\quad$ $y_t = g_y(W_y h_t + b_y)$

- **Sequence processing**
  can't be trained in parallel
- **Past information through hidden states**
  i strongly affects i+1 but influence is quickly lost in time
- **HARD TO TRAIN**
  as the path increases → the product chain increases
  Multiplications lead to vanishing (or exploding) gradients

What's wrong with RNNs?

$$\frac{\partial L}{\partial \mathrm{vec}(\mathbf{W})^\top} = \sum_{n=0}^{N} \sum_{k=0}^{n} \frac{\partial L_n}{\partial y_n} \frac{\partial y_n}{\partial h_n^\top} \frac{\partial h_n}{\partial h_k^\top} \frac{\partial h_k}{\partial \mathrm{vec}(\mathbf{W})^\top}$$

$$\frac{\partial h_K}{\partial h_0^\top} = \frac{\partial h_K}{\partial h_{K-1}^\top} \frac{\partial h_{K-1}}{\partial h_{K-2}^\top} \cdots \frac{\partial h_1}{\partial h_0^\top}$$

$h_t = g_h(W_h[h_{t-1}; x_t] + b_h)$ $\quad$ $y_t = g_y(W_y h_t + b_y)$

- **Sequence processing**
  can't be trained in parallel
- **Past information through hidden states**
  i strongly affects i+1 but influence is quickly lost in time
- **HARD TO TRAIN**
  as the path increases → the product chain increases
  Multiplications lead to vanishing (or exploding) gradients

~~SOLUTION~~ (WORKAROUND)

- Bi-directional models
- LSTMs

How to overcome CNN and RNN Problems?

How to overcome CNN and RNN Problems?

What if every word in a sentence could “talk” to every other word directly, regardless of the distance?

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Attention is all you need

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurip

… to attend to **all** positions in the decoder up to and including that position. **We need** to

… **We** implement this inside of scaled dot-product **attention** by masking out (setting to

☆ Save 99 Cite Cited by 216443 Related articles All 70 versions »

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Attention is **all you need**

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurip

… to attend to **all** positions in the decoder up to and including that position. **We need** to

… **We** implement this inside of scaled dot-product **attention** by masking out (setting to

☆ Save 99 Cite Cited by 216443 Related articles All 70 versions »

15 keypoints is all you need [PDF] thecvf.com

M Snower, A Kadav, F Lai… - Proceedings of the IEEE …, 2020 - openaccess.thecvf.com

Pose-tracking is an important problem that requires identifying unique human pose-instances and matching them temporally across different frames in a video. However, existing pose-…

☆ Save 99 Cite Cited by 26 Related articles All 9 versions »

GMML is all you need [PDF] arxiv.org

S Atito, M Awais, J Kittler - arXiv preprint arXiv:2205.14986, 2022 - arxiv.org

Vision transformers have generated significant interest in the computer vision community because of their flexibility in exploiting contextual information, whether it is sharply confined …

☆ Save 99 Cite Cited by 5 Related articles All 3 versions »

Rezero is all you need: Fast convergence at large depth [PDF] mlr.press

T Bachlechner, BP Majumder, H Mao… - Uncertainty in …, 2021 - proceedings.mlr.press

Deep networks often suffer from vanishing or exploding gradients due to inefficient signal propagation, leading to long training times or convergence difficulties. Various architecture …

☆ Save 99 Cite Cited by 190 Related articles All 7 versions »

Diversity is all you need: Learning skills without a reward function [PDF] arxiv.org

B Eysenbach, A Gupta, J Ibarz, S Levine - arXiv preprint arXiv:1802.06070, 2018 - arxiv.org

… In this paper, we propose "Diversity **is All You Need**"(DIAYN), a method for learning useful skills without a reward function. Our proposed method learns skills by maximizing an …

☆ Save 99 Cite Cited by 859 Related articles All 4 versions »

Hopfield networks is all you need [PDF] arxiv.org

H Ramsauer, B Schäfl, J Lehner, P Seidl… - arXiv preprint arXiv …, 2020 - arxiv.org

We introduce a modern Hopfield network with continuous states and a corresponding update rule. The new Hopfield network can store exponentially (with the dimension of the …

☆ Save 99 Cite Cited by 270 Related articles All 6 versions »

CNN is all you need [PDF] arxiv.org

Q Chen, R Wu - arXiv preprint arXiv:1712.09662, 2017 - arxiv.org

The Convolution Neural Network (CNN) has demonstrated the unique advantage in audio, image and text learning; recently it has also challenged Recurrent Neural Networks (RNNs) …

☆ Save 99 Cite Cited by 18 Related articles All 2 versions »

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Attention is all you need

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurip

… to attend to **all** positions in the decoder up to and including that position. **We need** to

… **We** implement this inside of scaled dot-product **attention** by masking out (setting to

☆ Save 99 Cite Cited by 216443 Related articles All 70 versions »

No convolutional operations

No recurrent units

Transformers and Attention

Transformers and Attention

Transformers and Attention

Transformer Encoder

- Input: **Tokens**
- Encodes context in **parallel**

Transformer Decoder

- Access to entire encoded context
- **autoregressive**

Transformers and Attention

Self-Attention

Transformer

- N Transformer Encoder Layers
- Input Embedding
- Positional Encoding
- Multi-Head Attention
- Add & Norm
- Feed Forward

Self-Attention (High Level)

The animal didn't cross the street because it was too tired

Self-Attention (High Level)

The animal didn't cross the street because it was too tired

What does "it" in this sentence refer to?

Self-Attention (High Level)

The animal didn't cross the street because it was too tired

Self-Attention (High Level)

The animal didn't cross the street because it was too tired

Query: Search bar
Keys: Database videos
Values: Best Matched videos

YouTube DK

transformers deep learning

All | Shorts | Unwatched | Watched | Recently uploaded | Live

About these results | Filters

Library
History

Sign in to like videos, comment and subscribe.

Sign in

Explore
Trending
Music
Films
Gaming
Sport

Browse channels

More from YouTube
YouTube Premium
YouTube Music
YouTube Kids

Deep Learning Course - Ranked #1 AI & ML Course

PG Program in AI & Machine Learning covers Python, Deep Learning, NLP & more. Enroll Now! Hands-on learning with Capstone...

Ad · https://www.simplilearn.com/ai-ml/pgprogram

Illustrated Guide to Transformers Neural Network: A step by step explanation

676K views • 3 years ago

The A.I. Hacker - Michael Phi

Transformers are the rage nowadays, but how do they work? This video demystifies the novel **neural network** architecture with ...

Intro | Input Embedding | 4. Encoder Layer | 3. Multi-headed Attention | Residual Connection, Layer... 8 chapters

What are Transformers (Machine Learning Model)?

166K views • 1 year ago

IBM Technology

Transformers? In this case, we're talking about a **machine learning** model, and in this video Martin Keen explains what ...

Subtitles

Why Did the Banana Cross the Road | Transformers Are a Form of Semi Supervised Learning |... 4 moments

Transformer Neural Networks, ChatGPT's foundation, Clearly Explained!!!

127K views • 1 month ago

StatQuest with Josh Starmer

Transformer Neural Networks are the heart of pretty much everything exciting in AI right now. ChatGPT, Google Translate and ...

Subtitles

Another analogy: The Cocktail party

Imagine you are at a crowded party

Looking for somebody to know about AI

You look around at everyone (Keys)

Find a match! Focus attention!
Extract information (Value)

Self-Attention

[Slides from Illustrated Transformer]

Self-Attention

[Slides from Illustrated Transformer]

Self-Attention

[Slides from Illustrated Transformer]

Self-Attention

[Slides from Illustrated Transformer]

Self-Attention

[Slides from Illustrated Transformer]

Multi-Head Attention

Multi-Head Attention

[Slides from Illustrated Transformer]

self-attention is permutation-equivariant

Discuss with your neighboor (2-3 minutes)

- **What is equivariant? What is invariant?**
- **Is self-attention permutation-equivariant? Why?**

self-attention is permutation-equivariant

Positional Encoding

Transformer

- N Transformer Encoder Layers
- Input Embedding
- Positional Encoding
- Multi-Head Attention
- Add & Norm
- Feed Forward

Positional Encoding

[Slides from Illustrated Transformer]

Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

[Slides from Illustrated Transformer]

Positional Encoding

PE can also be learnable! (nn.Parameter)

[Slides from Illustrated Transformer]

Transformer Encoder

Transformer

- N Transformer Encoder Layers
- Input Embedding
- Positional Encoding
- Multi-Head Attention
- Add & Norm
- Feed Forward

Text classification

Dette kursus er fantastisk

Text classification

Text classification

Text classification

1)Remove Transformer Decoder

2)Add avgpool at the Encoder

3)Add a simple Linear Layer as “decoder”

Text classification

1)Remove Transformer Decoder

2)Add avgpool at the Encoder

3)Add a simple Linear Layer as “decoder”

4)Add a [CLS] token and use its output to the Linear layer (no avgpool/ignore other ouputs)**

BERT (Bidirectional Encoder Representation Transformers)

Exercise 1

Sentiment classification of IMDB movie reviews

- Complete the implementation of a transformer model
- Train your model on the given dataset
- Test and evaluate your results
- Check out the effect of several choices
- Report your results (1-2 pages)

Exercise 1

1) Implement Attention class

2) Implement Positional Encoding

3) Train a transformer-based model for sentiment classification of IMDB reviews (positive/negative)

4) Hyper-parameter effect: Nlayers, Nheads, Dmodel, lr

5) OPTIONAL: CLS TOKEN, Learnable PE

Exercise 1

```python
class Attention(nn.Module):
    def __init__(self, embed_dim, num_heads):
        super().__init__()

        assert embed_dim % num_heads == 0, f'Embedding dimension ({embed_dim}) should be divisible by number of heads ({num_heads})'
        self.num_heads = num_heads
        self.head_dim = embed_dim // num_heads
        self.scale = self.head_dim ** -0.5

        ######################## insert code here ########################
        #
        #
        #
        ##################################################################
        self.o_projection = nn.Linear(embed_dim, embed_dim)
```

```python
def forward(self, x):

    batch_size, seq_length, embed_dim = x.size()
    keys    = self.k_projection(x)
    queries = self.q_projection(x)
    values  = self.v_projeciton(x)

    # Rearrange keys, queries and values
    # from batch_size x seq_length x embed_dim to (batch_size x num_head) x seq_length x head_dim
    keys = rearrange(keys, 'b s (h d) -> (b h) s d', h=self.num_heads, d=self.head_dim)
    queries = rearrange(queries, 'b s (h d) -> (b h) s d', h=self.num_heads, d=self.head_dim)
    values = rearrange(values, 'b s (h d) -> (b h) s d', h=self.num_heads, d=self.head_dim)

    ###################### insert code here ######################
    #
    #
    #
    #
    ##############################################################

    # Rearragne output
    # from (batch_size x num_head) x seq_length x head_dim to batch_size x seq_length x embed_dim
    out = rearrange(out, '(b h) s d -> b s (h d)', h=self.num_heads, d=self.head_dim)

    assert attention.size() == (batch_size*self.num_heads, seq_length, seq_length)
    assert out.size() == (batch_size, seq_length, embed_dim)

    return self.o_projection(out)
```

References

Attention is all you need: https://arxiv.org/abs/1706.03762

Illustrated Transformer: http://jalammar.github.io/illustrated-transformer/

Annotated transformer: https://nlp.seas.harvard.edu/2018/04/03/attention.html

Attention?Attention!: https://lilianweng.github.io/posts/2018-06-24-attention/

BERT paper: https://arxiv.org/abs/1810.04805

RNN cheatsheet: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Stanford CS25- Transformer United (seminar with 25 lectures)
https://www.youtube.com/playlist?list=PLoROMvodv4rNiJRchCzutFw5ItR_Z27CM

Questions???

Join at menti.com | use code 4955 5579

Like?

Unclear?