Deep Learning

Bundesbank Workshop on Deep Learning Day 2

Marcel Neunhoeffer, Christian Arnold & Sebastian Sternberg

29.03.2019

Introduction

Introduction

What's happened so far?

Your Expectations

You and this DL class

- Did you learn something new on Day 1?
- If so, what did you learn?
- What do you expect to take home from Day 2?

Deep Learning Workshop

Day 1: Introduction

- Motivation
- How does it work?
- Running your first models

Day 2: Advanced Applications

- Deep Learning for Sequences (Timeseries)
- Generative Deep Learning
- Recent Advances in Deep Learning

Today's Agenda

1. Introduction
   - What's happened so far?
   - Deep Learning for Sequences
2. Recurrent Neural Nets
   - Adding Memory to Neural Nets
   - More Sophisticated Memory Modules
3. Introduction to Generative Adversarial Nets.
4. Privacy and Deep Learning

Introduction

Deep Learning for Sequences

Sequences are everywhere

- Time series data
- Natural language text
- Audio signals
- Videos

Relevant Applications for Bundesbank

- Chakraborty and Joseph (2017): Forecast UK CPI inflation using different machine learning techniques
- Rönnqvist and Sarlin (2017): Study of financial risk: predictive model that is able to detect infrequent events based on text data
- Fischer and Krauss (2018): Financial market predictions: long short-term memory networks (more on that tomorrow)
- Lecun et al. (2015): Overview paper on deep learning in the "Nature"

A Look at Different NN architectures

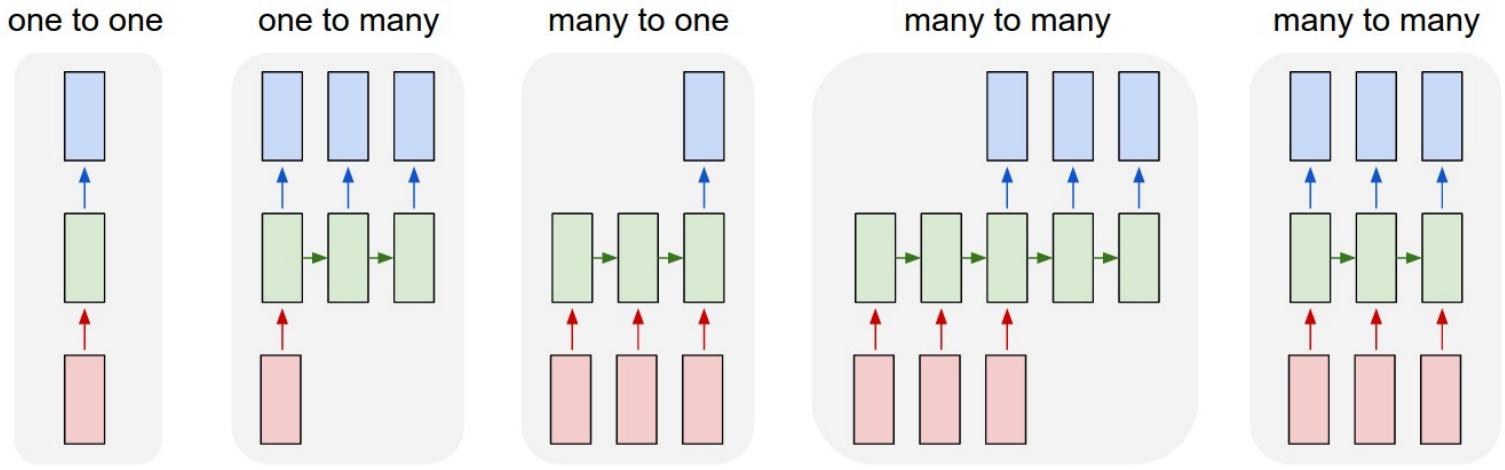


Figure 1: Different NN architectures. Source: Andrej Karpathy

- What architecture did we look at yesterday?
- How could we process sequences with that architecture?

The Jena Weather Dataset

- Weather timeseries data set recorded ate the Weather Station at the Max Planck Institute for Biogeochemistry in Jena.
- The data contains measurements of 14 different quantities (e.g. air temperature, atmospheric pressure, humidity, wind direction etc.), recorded every 10 minutes.
- The dataset contains all records from 2009 to 2016, a total of more than 420,000 observations.
- We will use it to build a model that predicts the air temperature 24 hours in the future.

A Common Sense Baseline

- Before running expensive models it is important to think of baseline models that we (ideally) want to beat.
- A common-sense approach for weather data is to always predict that the temperature 24 hours from now will be the same as the temperature right now.
- In the Jena dataset this method has a mean absolute error of 2.57°C.

A first Neural Net Baseline

- Let's see how well a neural network from yesterday would do at the task.
- We will go through all the necessary steps in the first code example.

Playtime

Code Nr. 1

How Do We Perceive Sequences?

As you are reading or listening to this sentence, you are processing it word by word while keeping memories of what came before; this gives you a fluid representation of the meaning conveyed by this sentence.

- We process information incrementally.
- We keep an internal model of what we are processing.
  - This model is built from past information,
  - and is constantly updated as new information comes in.
- A very simplified version of this can be added to neural nets.

Recurrent Neural Nets

Recurrent Neural Nets

Adding Memory to Neural Nets

Adding Memory to Neural Nets

- A recurrent neural net (RNN) adopts this principle.
- A RNN processes sequences by:
  - Iterating through the sequence.
  - Keeping a *state* that contains previous information.
- A RNN is a neural net that has an internal loop.

A simple RNN

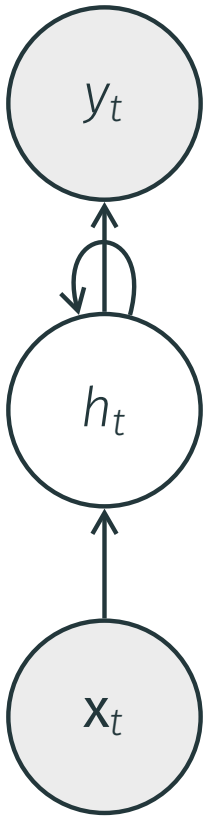


Figure 2: A simple RNN loops the output of h_t as an additional input to h_{t+1}.

Unrolling the RNN

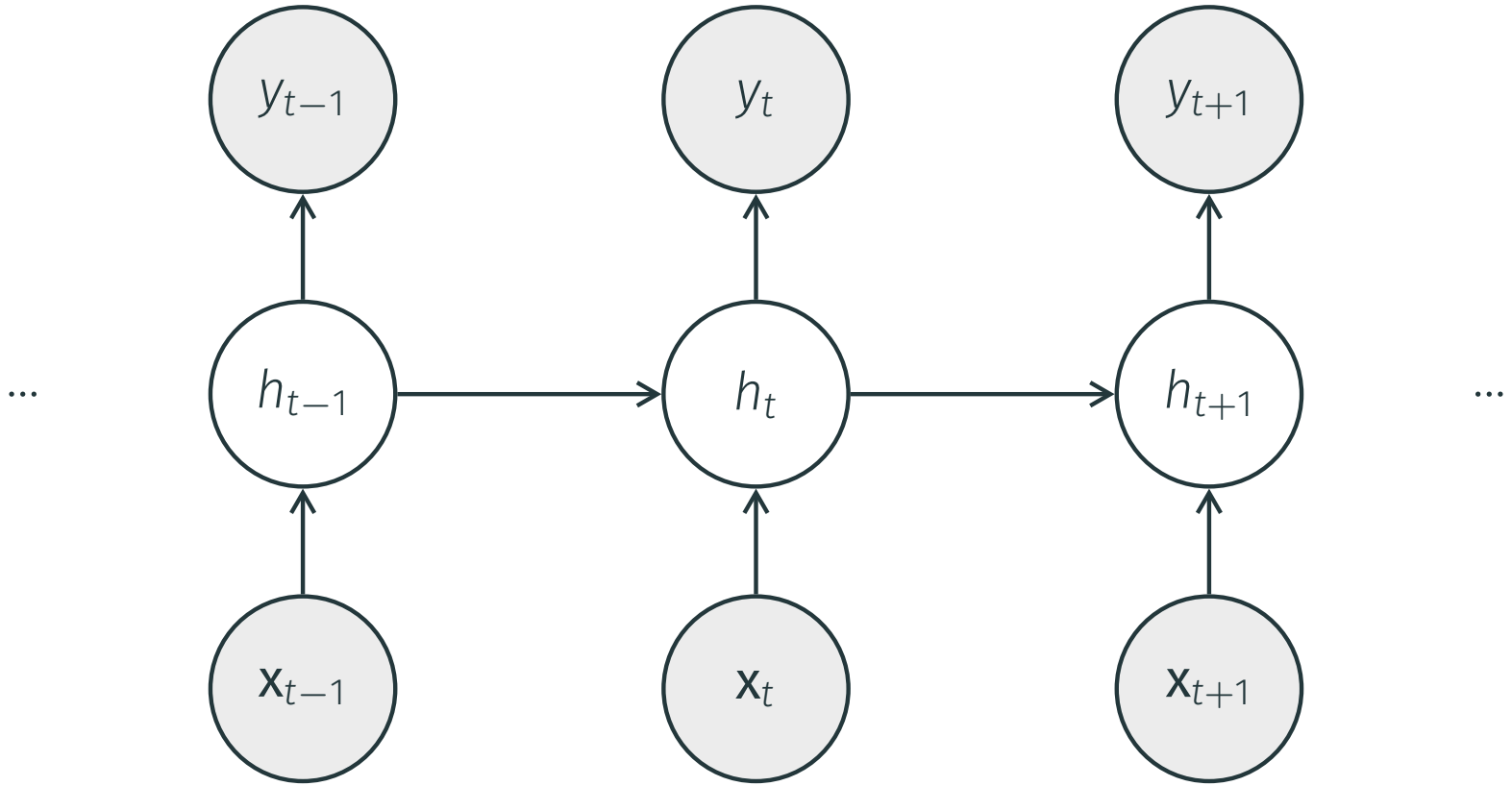


Figure 3: Unrolling the loop shows what happens in the RNN.

A closer look at h

- You can think of h as a shallow neural net.
- All that's new is that it now gets two inputs.
- $\mathbf{x}_t$ and the state of the previous time step h_{t-1}

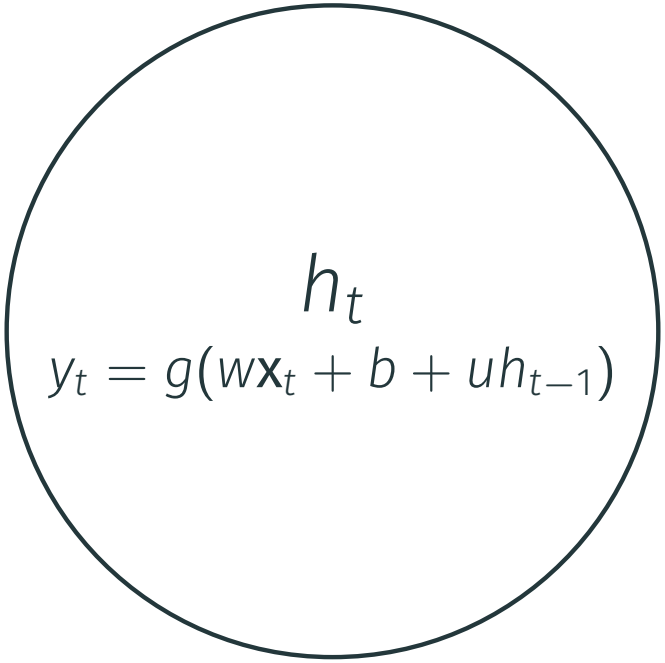


Figure 4: A look inside a RNN.

Understanding RNNs

- Unrolled RNNs are deep (in time).
  - They are deep even when the original network is shallow.
- *h* (the hidden state) serves two goals:
  1. Prediction: provide better features to the output layer.
  2. Memory: provide useful information to the next time step(s).

Simple RNNs tend to forget old information.

- In theory, this simple RNN architecture should allow the network to retain information about inputs it has seen many time steps before.
- In practice, it is impossible for simple RNNs to learn long-term dependencies.
- This is due to the vanishing gradient problem.

Simple RNNs tend to forget old information.

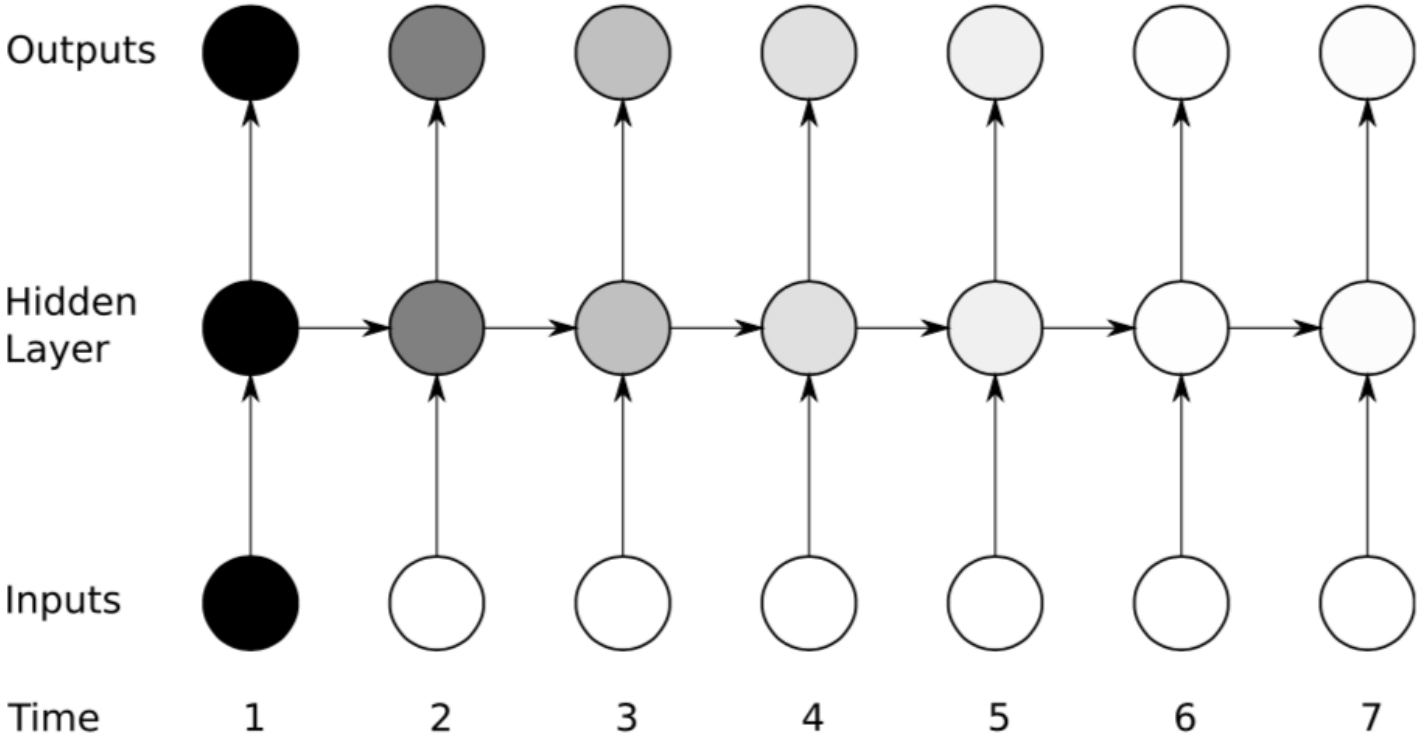


Figure 5: The vanishing gradient problem in RNNs. Source: Graves 2014.

Recurrent Neural Nets

More Sophisticated Memory Modules

Long Short Term Memory networks

Long Short Term Memory (LSTM) networks are an architecture for the hidden layer of a RNN.

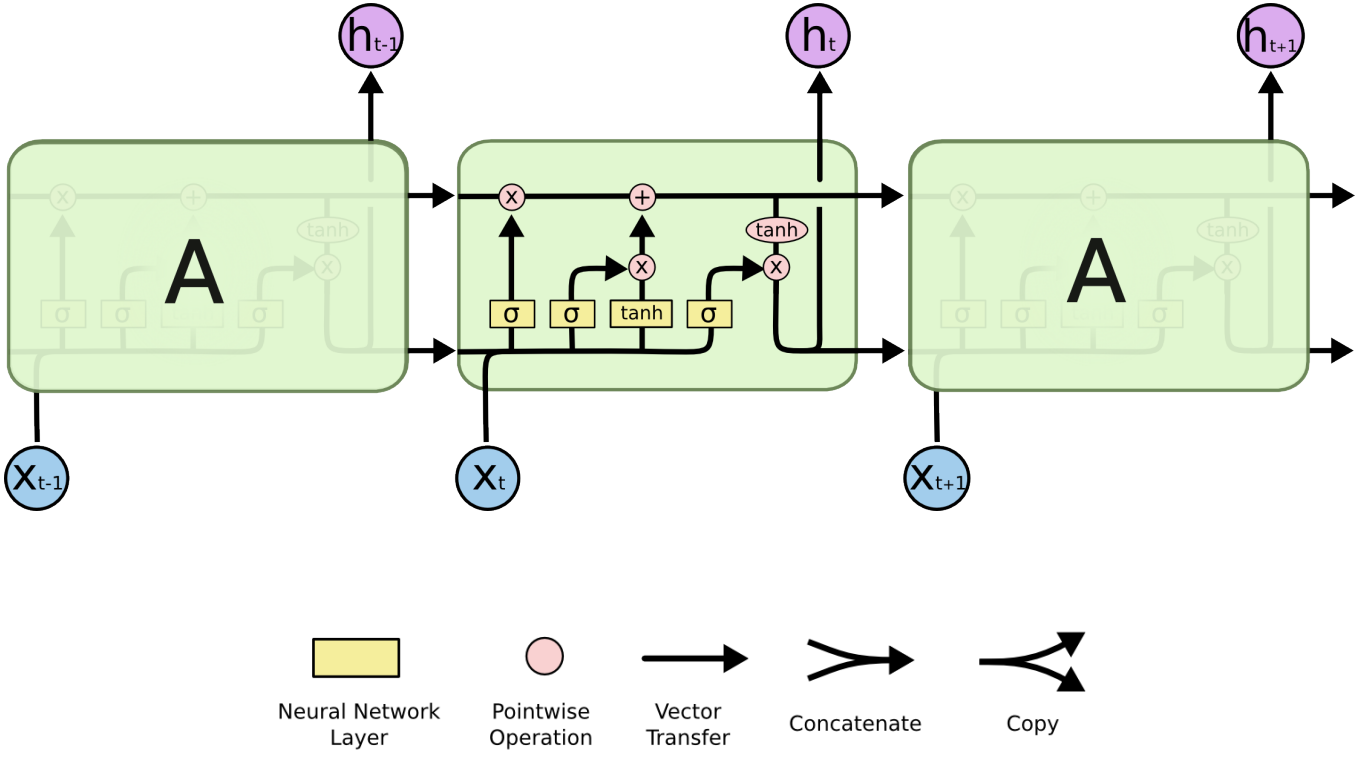


Figure 6: A LSTM cell in a RNN. Source: Olah 2015.

Long Short Term Memory networks

The cell state in a LSTM is designed to store long term information.

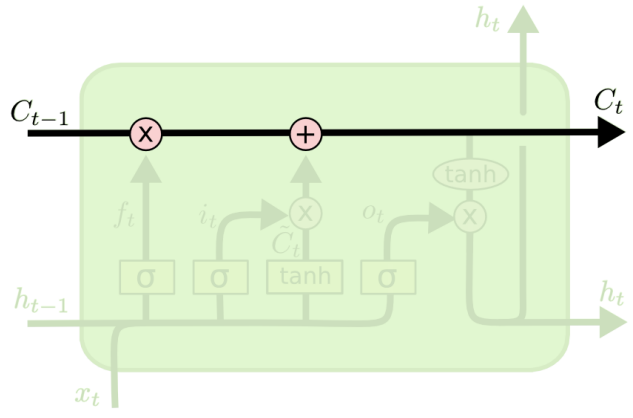
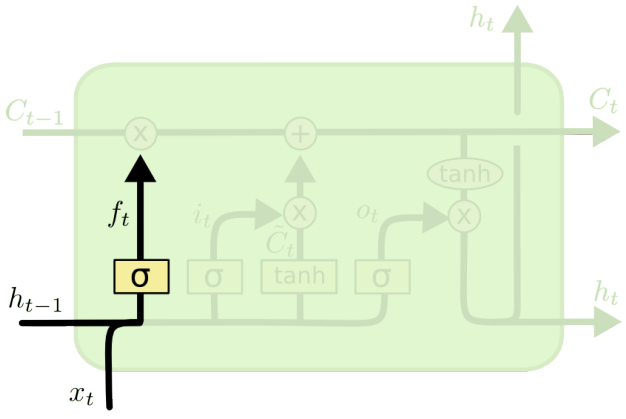


Figure 7: A LSTM cell in a RNN. Source: Olah 2015.

LSTM walk through - The forget gate

The forget gate controls to what extent the cell state is kept.

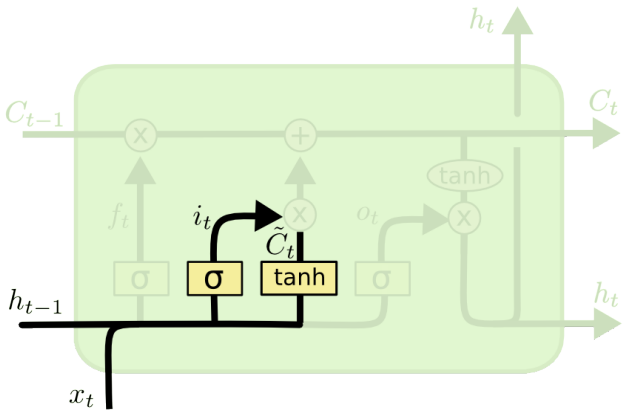


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 8: A LSTM cell in a RNN. Source: Olah 2015.

LSTM walk through - The input gate

The input gate controls to what extent the cell state is updated.

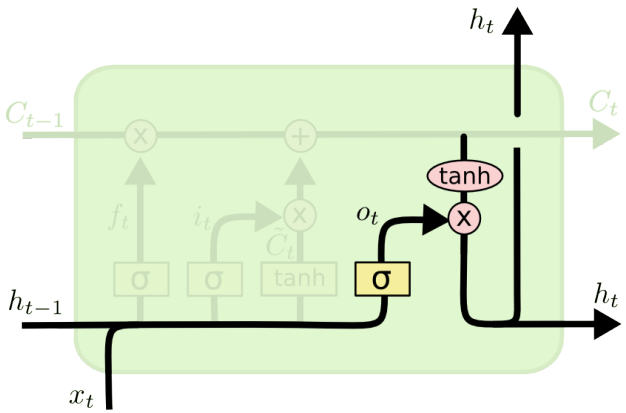


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 9: A LSTM cell in a RNN. Source: Olah 2015.

LSTM walk through - The output gate

The output gate decides what to output.

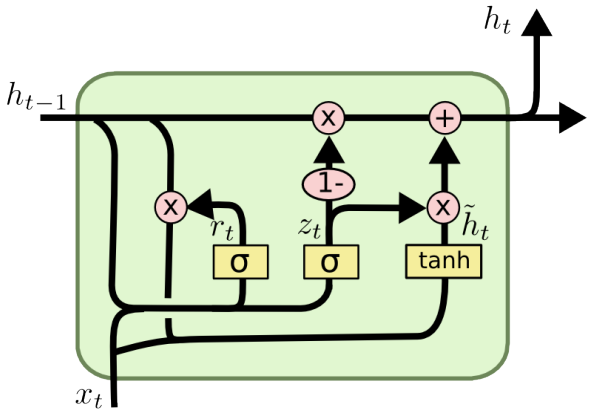


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 10: A LSTM cell in a RNN. Source: Olah 2015.

Variants of LSTM

Based on the architecture of LSTM many variants have been proposed. A recent and popular variant are so called Gated Recurrent Units (GRU).



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 11: A GRU cell in a RNN. Source: Olah 2015.

Playtime

Code Nr. 2

Words of Caution

- You might be inclined to use the models presented here to forecast the stock market or exchange rates or elections and so on.
- However, natural processes are usually easier to predict than social processes.
- In the case of markets or elections **only looking at past performance** is not a good predictor of future results.
- Still, if you have expert knowledge or additional data, testing these new DL models against the current models would be worthwhile.

Google Smart Compose

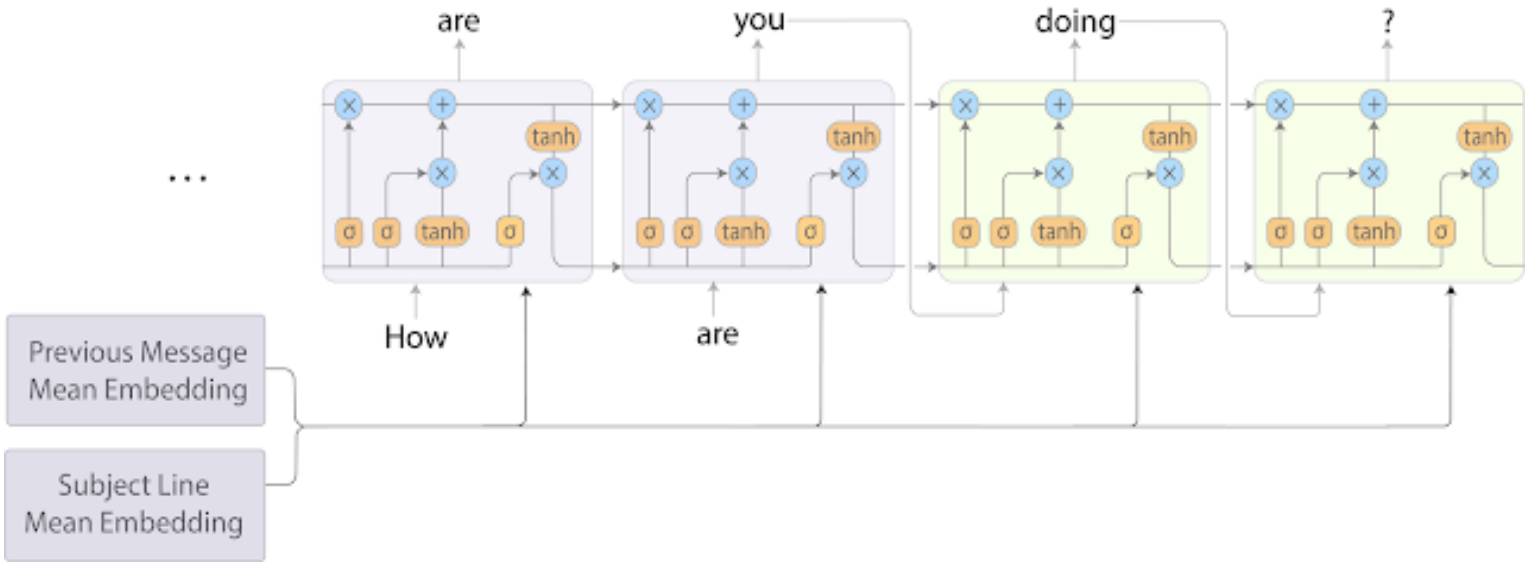


Figure 12: The memory module in Google Smart Compose. Source: Google AI Blog 2018.

Google Smart Compose in Action

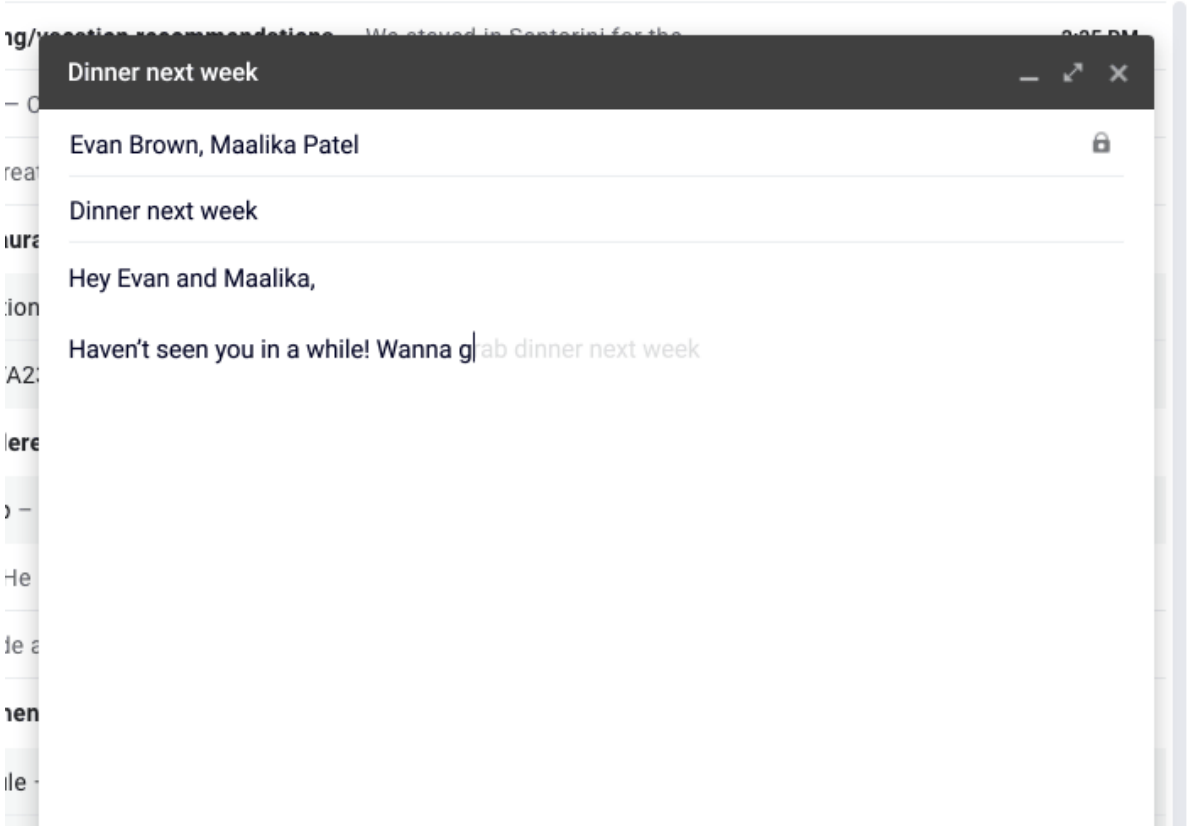


Figure 13: Text completion suggestions by Smart Compose. Source: Google AI Blog 2018.

Introduction to Generative Adversarial Nets.

Generative Adversarial Nets Can Make You Rich.

Figure 14: Edmond de Belamy, painted by a GAN. Source: Christie's.

Generative Adversarial Nets Are Surprisingly Simple.

- Generative Adversarial Nets (GANs), introduced by Goodfellow et al. (2014), allow it to sample from arbitrary joint (continuous) distributions.
- At its core, a GAN is a minimax game with two competing actors–a discriminator (D) trying to tell real from synthetic samples and a generator (G) to produce realistic synthetic samples from random noise.
- Formally, this two-player minimax game can be written as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\Big[\log D(x)\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[\log(1-D(G(z)))\Big]$$

where $p_{data}(x)$ is the distribution of the real data, X is a sample from $p_{data}(x)$. The generator network $G(z)$ takes as input z from $p(z)$, where z is a random sample from a probability distribution $p(z)$.

What Does A Schematic GAN Look Like?

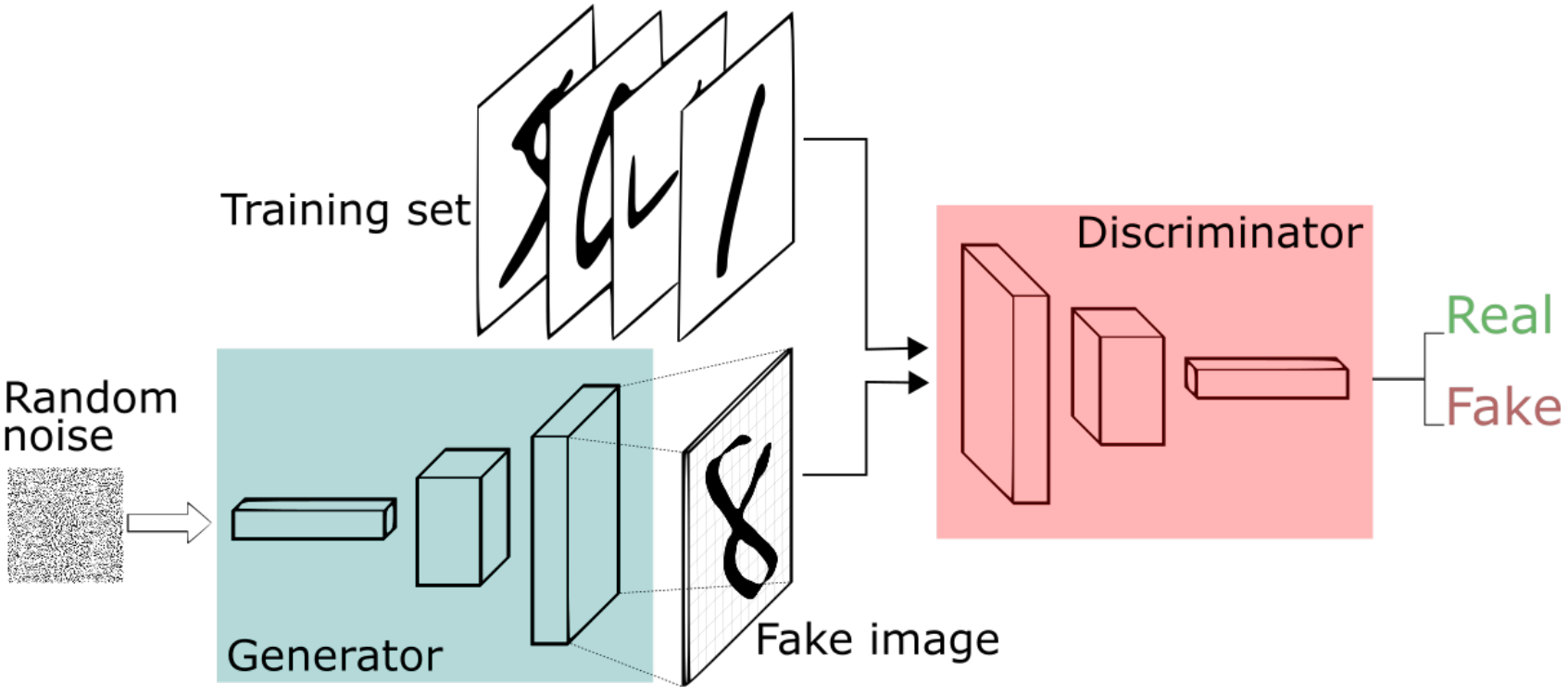


Figure 15: The architecture of a GAN. Source: Freecodecamp.org, Thalles Silva

A Closer Look at the Discriminator.

- The Discriminator is a binary classifier.
- An example for a well known binary classifier is a logit model.
- In GANs one will usually find neural nets.
- The Discriminator takes training examples and newly generated “fake” examples as input.
- The goal of the Discriminator is to tell real from “fake” examples.

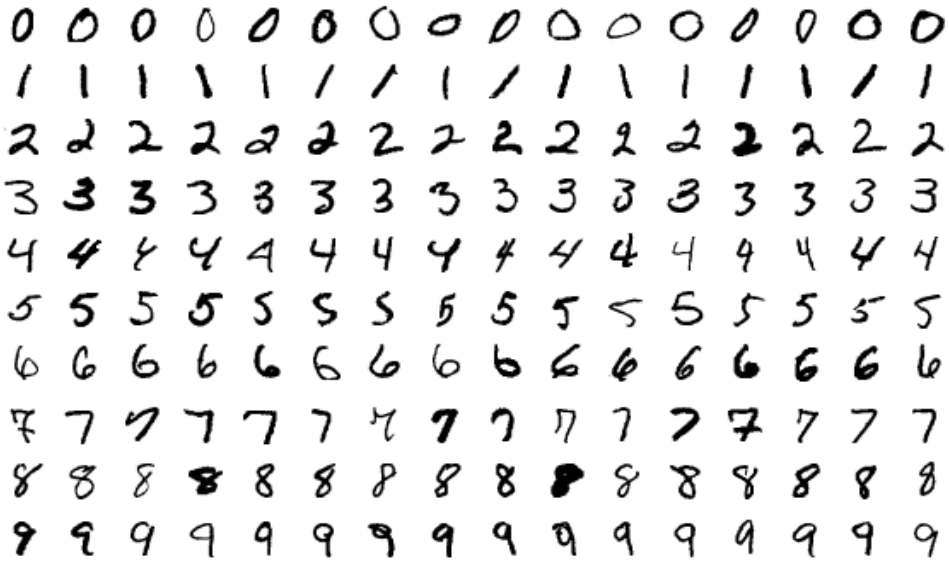
A Closer Look at the Generator.

- The Generator is a model that learns parameters to generate real looking examples from random noise.
- It takes random noise as an input and outputs “fake” data of the dimension of the original data.
- In GANs one will usually find neural nets with an output layer of the dimensions of the training data as a Generator.

How Can We Train a GAN?

- A GAN is a dynamic system.
- Every update of the Discriminator or Generator changes the optimization landscape.
- The goal is to achieve an equilibrium between the Generator and the Discriminator.
- A GAN is usually trained iteratively (update D, then update G, then D, then G, ...) with some form of mini batch Stochastic Gradient Descent (SGD).
- In practice it is hard to determine when an equilibrium between D and G is reached. However, sufficiently good Generators have proven to be powerful enough to produce impressive results (like paintings).

The MNIST Data Set Returns



- Modified National Institute of Standards and Technology database
- 60k training and 10k testing images of handwritten digits
- Black and white images from NIST normalized to fit into a 28x28 pixel box
- One of the CLASSIC machine learning data sets

Code Nr. 3

A Look at State of the Art Results.

Figure 16: Faces from a GAN by NVIDIA. Source: Karras, Laine and Aila 2018.

A Look at State of the Art Results.

Figure 17: Cars from a GAN by NVIDIA. Source: Karras, Laine and Aila 2018.

Privacy and Deep Learning

Privacy versus Sharing Data

Scientists, Companies and Agencies Often Use Privacy-Sensitive Data

- Surveys
- Data from Companies (e.g. Facebook)
- Government data

Trade-Off between Privacy and Data Sharing

- Sensitive data has to be protected.
- Data and results need to be shared (replication, knowledge transfer).

New (?) Risks to Privacy

- More and more data available–higher risk of linkage/differencing attacks.
- Successful model inversion attacks exposed sensitive information from model parameters.

Figure 18: Model Inversion Attack recovering images. Source: Frederikson, Jha and Ristenpart 2015.

Differential Privacy

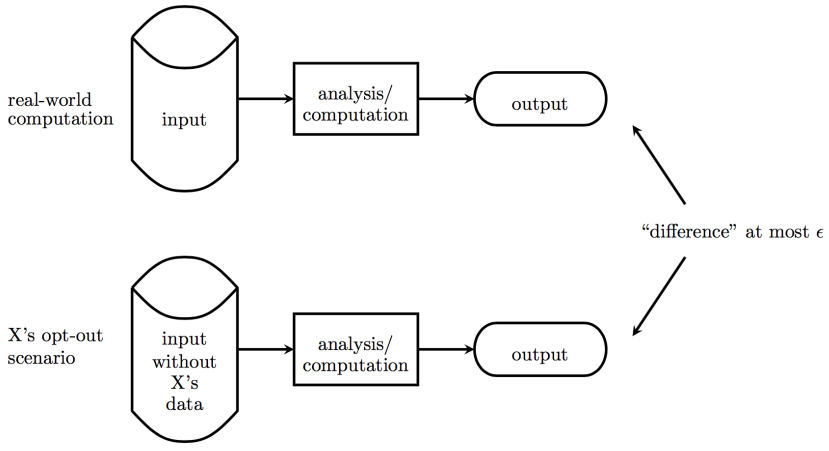


Figure 19: Source: Nissim et al. 2017.

- The result of a differentially private analysis reveals the same inference about any individual's private information, whether or not that individual's private information is included in the input to the analysis.
- Differential privacy allows for mathematical formulation (Dwork 2006, Dwork and Roth 2013)

(ϵ, δ)-Differential Privacy

Privacy Loss Parameters ϵ and δ

- ϵ quantifies how much information can be learned about any individual included in x, in a worst case scenario of an attacker with arbitrary side knowledge and arbitrary computational power.
- Small $\epsilon \rightarrow$ small potential privacy loss
- Larger values of ϵ mean that the potential privacy loss is higher, with ϵ = infinity meaning that all the information about any individual can be learned (i.e. by publishing the original data set).
- If $\delta = 0$, it is called pure or ϵ-differentially private data

(Differential Privacy in Practice

At the Whiteboard

- How does differential privacy work?

Differential Privacy in Practice

Implementing Differential Privacy in Neural Nets

- Basically you only have to add (the optimal amount of) noise to the gradient updates during training of a neural net.
- Since gradients can be unbounded usually the first step is to clip the gradients to some norm.
- Then the optimal noise for (ϵ, δ)-Differential Privacy is given by: $\frac{1}{\epsilon}\sqrt{2 \ln 1.25/\delta}$.
- If you want to learn more have a look at what tensorflow announced three weeks ago.

Appendix

Sources

Books

Chollet, François and J.J. Allaire. 2018. *Deep Learning with R*. Manning Publications.

Goodfellow, Ian and Yoshua Bengio and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Internet Resources

Ng, Andrew. *Deep Learning Specialization*. coursera.org.

Deep Learning Papers Reading Roadmap. github.com.

References

Chakraborty, C. and Joseph, A. (2017). Staff Working Paper No . 674: Machine learning at central banks.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning.

References II

Rönnqvist, S. and Sarlin, P. (2017). Bank distress in the news: Describing events through deep learning. *Neurocomputing*, 264:57–70.