FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Natural Language Generation

The State of the Art and the Concept of Ph.D. Thesis

Jakub Sido

Technical Report No. DCSE/TR-2020-05
September 2020
Distribution: public

Technical Report No. DCSE/TR-2020-05
September 2020

Natural Language Generation

The State of the Art and the Concept of Ph.D. Thesis

Jakub Sido

Abstract

Computational systems use natural language for communication with humans more often in the last years. This work summarises state-of-the-art approaches in the field of generative models, especially in the text domain. It offers a complex study of specific problems known from this domain and related ones like adversarial training, reinforcement learning, artificial neural networks, etc. It also addresses the usage of these models in the context of non-generative approaches and the possibility of combining both.

This work was supported by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

Copies of this report are available on
`http://www.kiv.zcu.cz/en/research/publications/`
or by surface mail on request sent to the following address:

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Contents

Chapter 1

Introduction

Computational systems use natural language for communication with humans more often in the last years. With growing computational capabilities and promising results, extensive research of neural network area has been possible. Researchers had explored various approaches to text generation, and neural generators show great success in many disciplines. Abstractive text summarization, machine translation, and image caption generation have shown great improvement during the last years.

We should not also omit natural language description generation from structured data like weather forecasts, short stock news or automatic systems providing information about departures or traffic.

In all of these tasks, natural language generation is a crucial step.

With the fastly growing amount of raw data in the world and the possibility of simulating whole problem space, unconventional attitudes to semi-supervised learning are getting more common. Approaches like reinforcement or adversarial learning open a wide area for applications in diverse domains and improve final results in many domains.

First, basic principles are introduced in this thesis. On this fundamental principles, more complex models are established.

Current trend in natural language processing is to use neural networks in various areas, and neural networks hold the state of the art in majority of tasks. The topic of this work is a research of applicable methods for text generation using the neural networks.

Chapter 2

Neural Networks

2.1 Artificial Neural Networks – The Historical Context

The first artificial neuron was introduced in 1943 by Warren McCulloch and Walter Pitts. The mathematical model called Linear Threshold Unit (LTU) was designed to model the process that takes place between neurons in the brain. This mathematical model has multiple boolean inputs and a single boolean value as an output. It was presented without any algorithms for training.

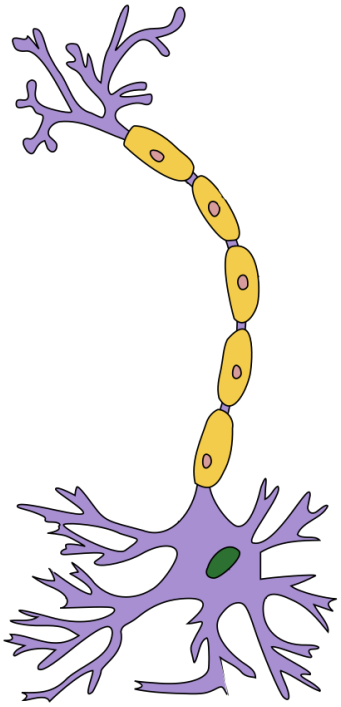


Figure 2.1: Biological neuron. (Image from Mankinds2020)

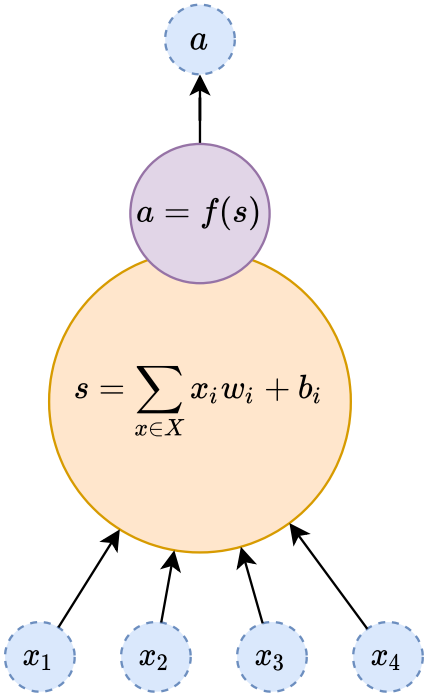


Figure 2.2: Mathematical model of the artificial neuron. w is the weight, b is the bias, x is the input and f is a nonlinear function.

It took 15 years for LTU to be adapted for a new, intentionally developed learning algorithm and used by Frank Rosenblatt in 1958 in his first model used in practise – the *perceptron*. It is also called single-layered perceptron, was designed for image classification.

Nonetheless, this simple model was not able to model even slightly more complex functions like non-equivalence; this ability was achieved by creating the so-called multi-layered perceptron in the next years with continuous activations instead of binary ones.

One layer of this multi-layered perceptron became an elemental block for building bigger models and this mechanism still remains in modern architectures. However, it is usually used for modelling vector transformation (with more than one neuron in the output layer) (Figure 2.3) unlike the original idea – binary classification.

In the matrix calculus, one layer of such a network can be written as:

$$\mathbf{A} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{2.1}$$

where W is a weight matrix, b a bias vector, x an input vector, and σ is a nonlinear activation function.

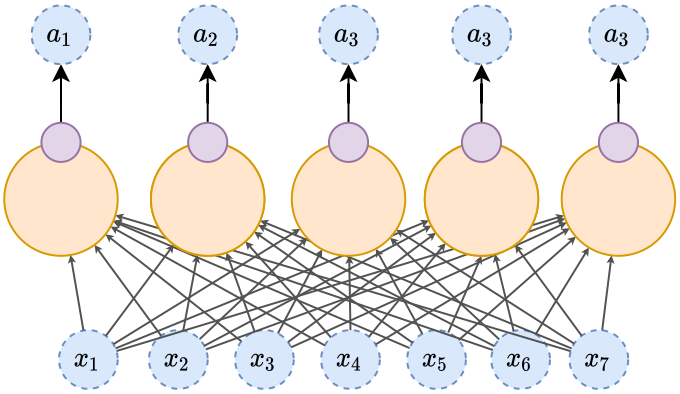


Figure 2.3: One layer composed of by perceptron cells

Nowadays, scientific works often use these layer as elementary blocks to build up from. These blocks are called dense or fully-connected layers and in diagrams they are usually depicted as simple boxes with a number of dimensions – number of single neurons.

2.2 Text Representation in Artificial Neural Networks

In machine learning, various ways of text representation have been used in the past.The Following chapter summarises mainstream approaches in natural language processing.

2.2.1 Atomic Text Units

Firstly, we have to realise that a written text could be split at different levels. We can use single characters; larger groups of characters split equidistantly such as n-grams; use some subword-based approach like prefixes, suffixes, etymons; or build own dictionary based on frequentistic analysis of a big amount of texts [Luong et al., 2013].

These atomic units can be represented in many ways. Probably the easiest way is to use *the one-hot*[1] representation. However, this representation is not convenient for capturing any relations among words. Synonyms can not be captured in this model in any other way than as enumeration on a higher level.

Due to this, projects like WordNet [Miller, 1998] have started. WordNet is a database of words created predominantly by humans. It contains mostly the information about super-subordinate relations. However, a wide scale of other relations is included too. Nonetheless, this approach falls behind the *distributional approach*.

The distributional approach is based on the idea that words are similar if they appear in a similar context [Harris, 1954].

[Mikolov et al., 2013] introduced a new, more straightforward way to train neural network language model. Thanks to the more efficient model, they were able to process huge amount of text which led to an interesting finding – the vectors (*embeddings*[2]) which arose from this model weree able to capture a wide scale of semantics. After this success, semantic spaces built up in this way or a similar one have been widely used in majority of systems. [Faruqui and Dyer, 2014] improved these models using multilingual correlations. Late approaches like *fastText* use the same mechanisms at the character level [Bojanowski et al., 2017]. Word embeddings trained on more specific tasks will capture task-specific features as well as relations between them – in a semantic space created by this method, contribution to the sentiment of each word will be captured in tasks like sentiment classification [Tang et al., 2014]. We also know models producing contextual dependent embeddings for words (see Sec. 2.5.2 on pg. 15).

2.2.2 Representation of Word Sequences

One of the biggest problems in text processing performed by neural networks, is the different length of input sequences. There are a few known practices for addressing this issue. We can use basic approaches like bag-of-words or bag-of-ngrams that hold every sequence in a vector of the same length. However, we would lose valuable information about relations among words which is hidden in their order. Because of this, it is not the usual way to process text in present-day systems. Using recurrent neural networks (RNNs) is more common because they do not suffer from different lengths of inputs. Nonetheless, they have other disadvantages caused by longer texts. RNNs are more deeply discussed in the following chapter. For approaches based on convolutional neural networks, some kind of alignment[3] is usually used. For short sequences, we can also use a simple addition of particular word vectors[4]. A very new way of representing sentences was introduced

[1] One-hot representation is a vector with one on the index of a specific word from a vocabulary and zeros elsewhere.

[2] Embedding can be explained as mapping discrete/categorical variables to a vector of real numbers.

[3] Adding some *zero* values at the end of short sequences to make them of the same length.

[4] With longer sequences, the sum will be closer to average and will not catch important information.

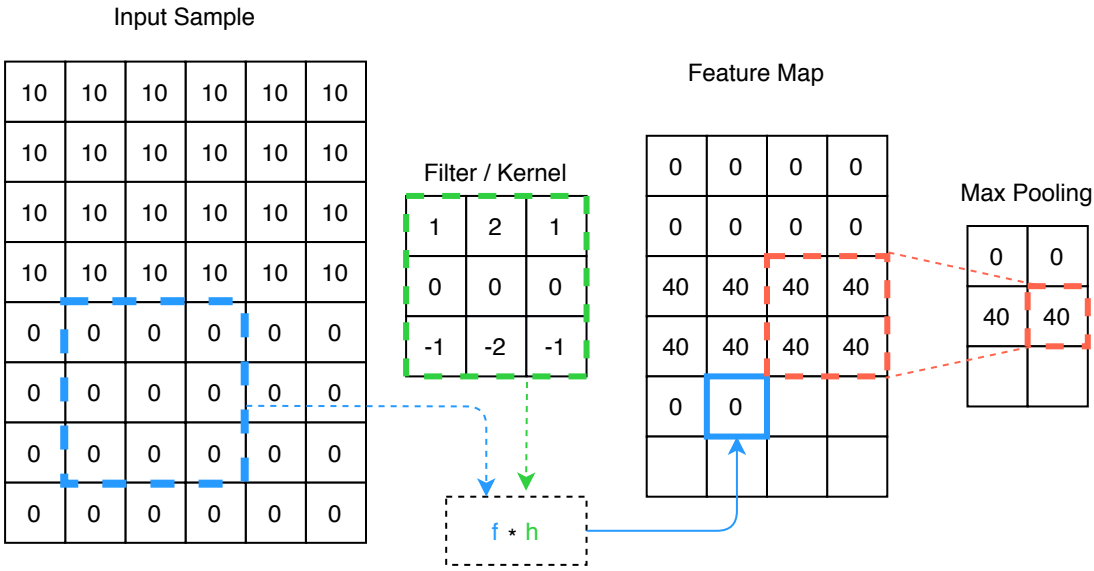


Figure 2.4: An illustration of basic principles – convolution and max-pooling. Convolution is not the standard matrix multiplication despite being often similarly denoted (Eq. 2.2).

recently. The basic idea is to train models on generic tasks and benefit from the unsupervised setup (language modelling, next sentence prediction) or to use a combination of a wide scale of different tasks. In the next sections (2.4.1, 2.5.1, 2.5.2, 2.5.2, 2.5.2), the most significant models are introduced.

2.3 Deep Convolutional Neural Network – DCNN

Convolutional neural networks, historically also called *time-delayed* networks (1D convolution), are often used in NLP nowadays. The main advantage of the convolutional approach lies in the possibility of computational parallelism – operations in one convolutional layer are independent. [Lang et al., 1990].

CNN architectures consist of convolutional layers that compute discrete convolution between a part of an input and a filter (often also called *kernel*) for each possible position of the filter in a sample (Eq. 2.2). A *feature map* is the result of this process and it captures all values of the convolution for all possible positions of the filter.

$$(f * h)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x - i, y - j) \cdot h(i, j) \tag{2.2}$$

For a better generalisation capability and a higher level of abstraction on higher layers, the convolutional layer is followed by a pooling layer which reduces the high dimensionality to preserve the main features (Figure 2.4). We know lots of different pooling types and each of them has a specific use (Table. 2.1). However, in the current deep approaches, *max pooling* is the first-choice instrument. *Max-over-time* pooling is a particular case of max-pooling over sequences where the maximum of the whole sequence is taken which is widely used in sequence classification tasks.

Pooling type	Meaning
Max pooling	Is this feature present anywhere in the range?
Max over time pooling	Does the sample contain this feature?
Average pooling	How prevalent is this feature over the entire range?
k-Max pooling	Is this feature present up to k times?
Dynamic pooling	Is this feature at the beginning or at the end?

Table 2.1: This table shows some examples of different pooling types. Each of them has a specific usage and the proper choice is often task-dependent.

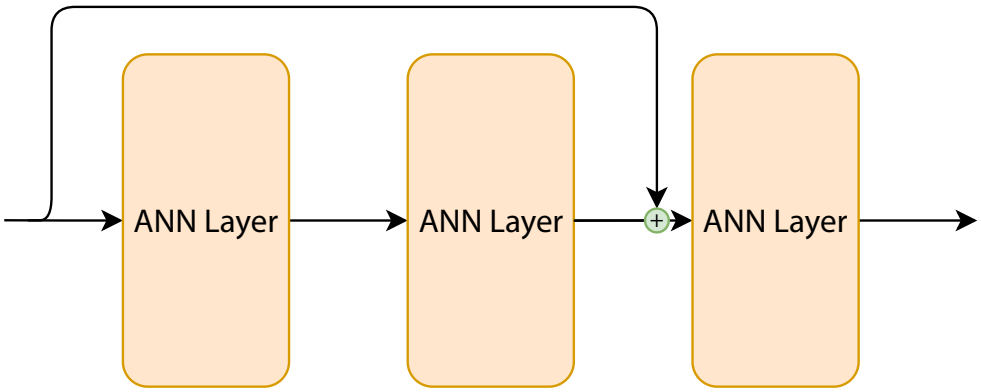


Figure 2.5: Residual connection

Note 1: Vanishing Gradient

Vanishing gradient is a problem which became serious issue in bigger and deeper models. Because of the deep architecture, the gradient propagated through the model becomes smaller[a] with increasing depth. There are good practises that more or less suppress this issues in different models and setups – the so-called residual/highway connection (Figure 2.5), scalled dot-product (see Sec. 2.5.1 on pg. 14) in attention mechanism or Glorot uniform initialisation [Glorot and Bengio, 2010].

[a] It can sometimes be even bigger which leads to similar problem called *exploding gradient*.

With the increasing computational capabilities, deeper architectures are often used which brings the issue of **vanishing gradient** (See Note 1 on page 5) [Bengio et al., 1994]. There are lots of methods dealing with this problem across a wide scale of architectures. However, using *residual connections* is probably the most common one(Figure 2.5). The main idea of the residual network (ResNet) is to make shortcuts for gradient propagation [Tai et al., 2017].

[Dauphin et al., 2017] add *an attention mechanism* (see Sec. 2.5) to a CNN architecture in the language modelling task.

In deeper architectures and GAN[5] architectures, larger residual blocks consisting of convolution, batch normalisation, and activation are often used for better training stability [Jin et al., 2017].

Siamese Network [Bromley et al., 1993] have introduced a new model for sample similarity (Figure 2.6). It uses two time-delayed neural networks for transforming basic features of two hand-written signatures into feature vectors and measures the distance between them. [Mueller and Thyagarajan, 2016] use the same principle for the sentence similarity task. [Yin et al., 2016]

[5] Generative adversarial network – see 3.3.1

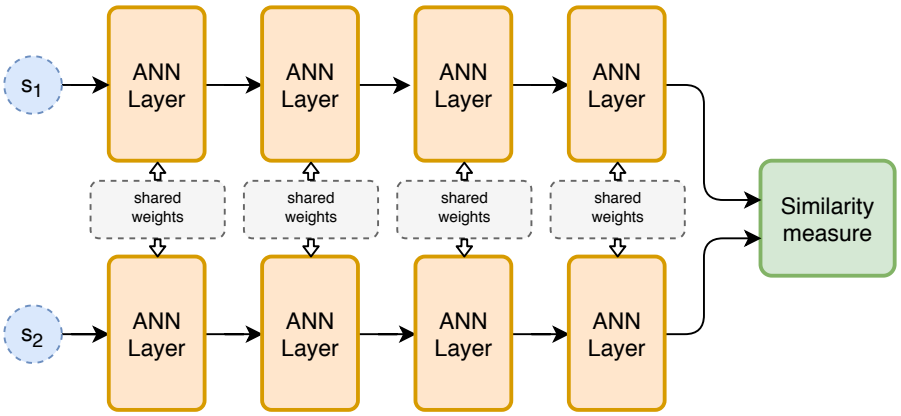


Figure 2.6: Schematic visualisation of the Siamese architecture

add an *attention mechanism* to these *siamese networks*.

Nowadays, *siamese networks* are widely used in similarity tasks in image processing [Varior et al., 2016, Baraldi et al., 2015] and text processing [Neculoiu et al., 2016, Baziotis et al., 2017].

CNN for sentence representation There is one main approach to create any sentence representation with convolutional architectures – in the text domain, a simple model with only one layer with filters of different sizes with max-over-time pooling can be used to reach exciting results (Figure 2.7) [Sido et al., 2019b]. With this setup, we get an unordered n-gram-based sentence representation. This straightforward approach would not work well on simple one-hot representations. However, we can get more interesting results with the same model if we use a semantic vector space. In such case, we bring the possibility of modelling the similarity of words. Then, the filters model ordered sequences of semantically specific tokens. This sentence representation can be used for text classification tasks or as an encoder part of any other application. Besides, we can build up a deeper model with residual connections to make the model capable of capturing more complex patterns.

In image processing, [Dai et al., 2015] use DCNNs for semantic segmentation, object detection or image classification.

2.4 Recurrent Neural Networks – RNN

The basic idea of RNNs is to use the output as the input into the next step. The next step could mean next time step in a time series or logically succeeding data in some other meaning; e.g. the next image in a video sequence, a next column of pixels in an image processing going from left to right, or a next word in a text.

Recurrent neural networks are often used for sequences of unknown lengths. They are often aligned on the same length for optimisation and better parallelisation; in principle, however, RNNs do not need the input to be of the same length.

For a better picture, we can imagine a recurrent unit as more stacked units using outputs from previous time steps as own inputs while sharing parameters (Figure 2.8).

Vanishing Gradient Problem The vanishing gradient is a big issue in standard RNNs. An error propagation across all weights through the whole sequence is problematic because of the long

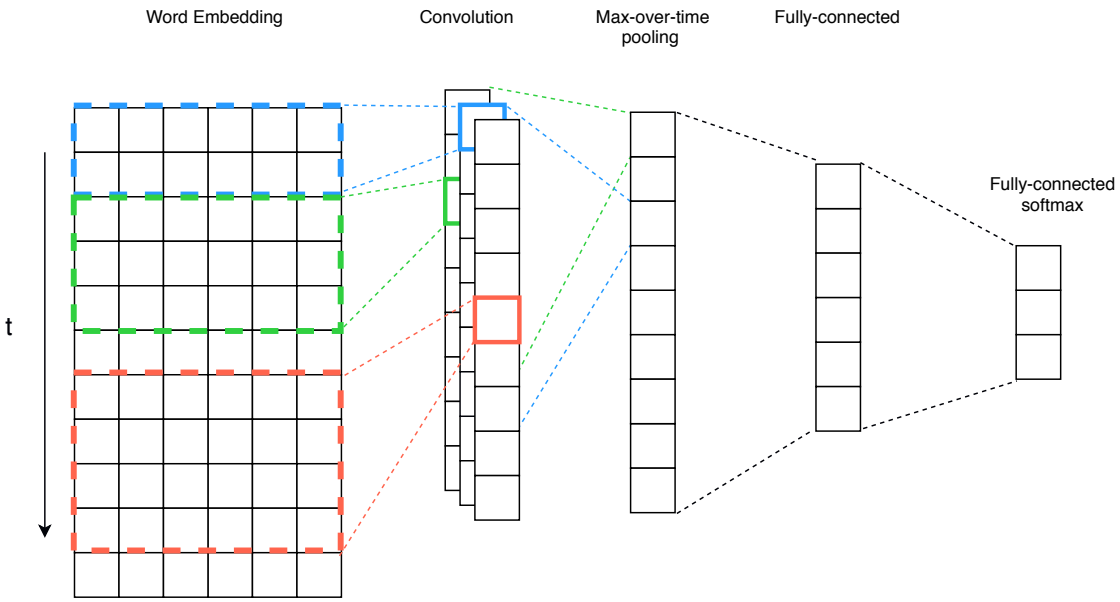


Figure 2.7: Basic CNN model for a text classification [Sido et al., 2019b].

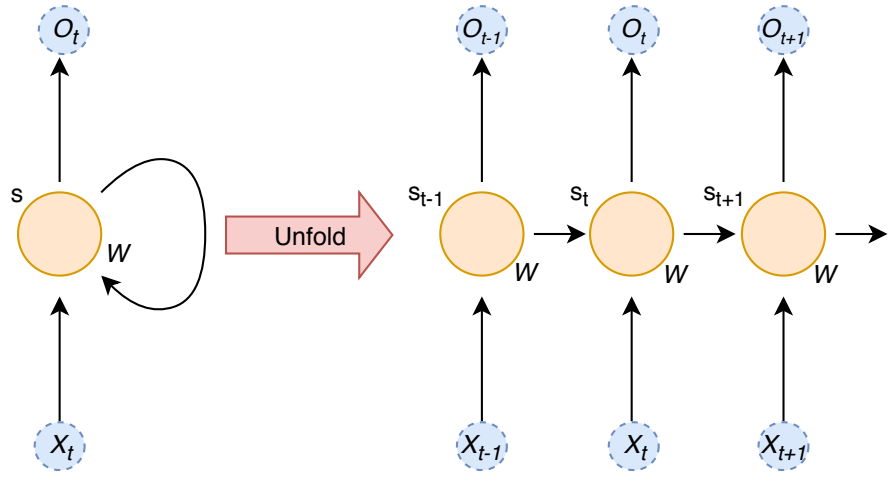


Figure 2.8: Basic RNN cell unfold

transformation chain from the input into the output.

A gating mechanism is often integrated into recurrent units for better performance. It allows better gradient propagation back through the time [Chung et al., 2014]. Currently, the most used gated recurrent units are LSTM – Long Short Term Memory or GRU – Gated Recurrent Unit. Both are used in many variations.

Long-short-term-memory – LSTM [Hochreiter and Schmidhuber, 1997] This unit is based on several neural gates (input, forget, output) that change the inner states of the cells. The following equations describe projection of the input x_t onto the output h_t. The same transformation is schematically depicted on Figure 2.9.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.3}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.4}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.5}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.6}$$
$$h_t = o_t \circ \sigma_h(c_t) \tag{2.7}$$

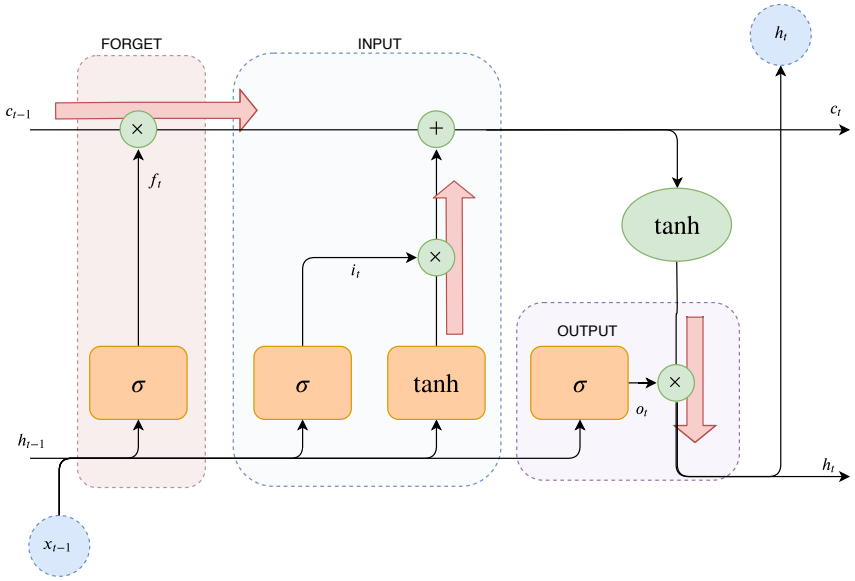


Figure 2.9: Scheme of the LSTM unit

Initial values for cell state c_0 and hidden state h_0 are zero vectors. The symbol $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time steps. Lower index f is for the forget gate, i for the input gate, o for the output gate. W and U are weight matrices.

Gated Recurrent Unit – GRU GRU is another new architecture of the recurrent unit [Cho et al., 2014]. In contrast to the LSTM, the cell state is not used here; the different gating system leads to similar results. Some studies show a different performance of GRU and LSTM on some tasks, none of them proves superiority of any of them in performance (regarding the model accuracy) – the results are task-dependent. Also, a non-negligible improvement of time efficiency is

observable in several tasks [Cho et al., 2014]. So far, we can not do any general conclusion about their propriety.

The following equations describe the projection of the input x_t onto the output h_t. The same transformation is schematically depicted on Figure 2.10. The initial value for the hidden state h_0 is zero vector.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.8}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2.9}$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{2.10}$$

W and U are weight matrices. Lower index z is for the update gate and r for the reset gate.

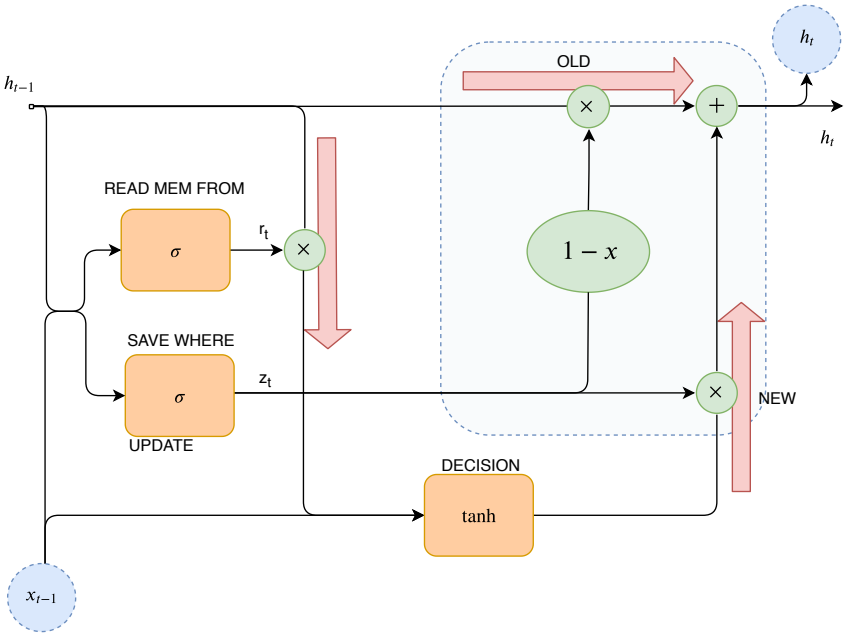


Figure 2.10: Scheme of the GRU unit.

Char-based RNN Different levels of atomicity were mentioned before. A lot of recent approaches use characters not only for language modelling [Kim et al., 2016] but even for semantic analysis [Chiu and Nichols, 2016]. Instead of using embeddings for words, we can train character-embeddings and search groups of characters and their dependencies [Ling et al., 2015].

Bidirectional RNN In order to get both, the left and the right context of some word in a sentence, we can use the bidirectional RNN [Schuster and Paliwal, 1997]. This technique is widely used for tasks where we need to classify each time step and both contexts are related, like e.g. in speech recognition [Graves et al., 2013], relation classification [Zhou et al., 2016], or sequence tagging [Huang et al., 2015].

Stacked RNN As the deeper CNN can take into account a larger context and more abstract primitives in the input, we can profit from the deeper architecture of RNN layers as well. However, intuition about what is happening is not so clear like in the case of the deep CNN. The CNNs preserve the relative position of the primitives detected at lower levels, dimensions are reduced, and in higher layers, we observe high-level features; also the surrounding context starts on a small

neighbourhood and higher layers take into account larger areas. There are lots of visualisation of this [fea, 2017]. In the RNN architectures, dimensionality reduction is not usual. Modern models use stacked layers of the same size [Prakash et al., 2016] with significant improvement of accuracy. Also, the whole context is processed on the first layer in the RNN architectures (right and left – unconditioned by each other) and on the second layer, the left context can be conditioned even on the right context and vice versa. Despite all of that, many papers have shown profit from deeper RNN architectures.

2.4.1 ELMo – Embeddings from Language Models

[Peters et al., 2018] introduced a more universal way of creating embeddings for cross-task usage. *ELMo word representations* are functions of the entire input sentence. They use two-layer bidirectional language models[6] with character level convolution for word representation and use weighted task-specific addition of each layer. They profit from stacked recurrent architecture and their representation catches different features on different levels (syntactic, semantic) fine-tuned for a specific task. The best published model has two bidirectional LSTM layers with 4096 units and dimension of 512 in the projection layer. The context insensitive representation type uses 2048 character n-gram convolutional filters, linearly projected down to 512; in total 93.6 million parameters. The model is public [7].

2.5 Attention Mechanism

Attention mechanism for RNN was introduced by [Mnih et al., 2014] on an image classification task. The next researchers use the same mechanism in the neural machine translation task [Bahdanau et al., 2014]. It is a first application of attention in natural language processing. The main idea is to make a shortcut in the standard encoder-decoder approach (see Sec. 3.5.1 on pg. 35) and bring text processing closer to humans. When people read a text, they do not act according to the standard encoder-decoder scheme. We are turning back for some pieces of information into the source sentence during the translation [att, 2016].

We do it this way when using attention. When generating word in the decode phase, we use every single word in the source sentence instead of using only the hidden state of the decoder.

The hidden state of the decoder is computed as follows:

$$s_t = f(s_{t-1}, y_{t-1}, c_t), \tag{2.11}$$

where s_{t-1} is the previous hidden state, y_{t-1} is the previous output of the decoder, and c_t is the context vector which is computed as follows:

$$c_t = \sum_{i=1}^{n} \hat{a}_{t,i} h_i. \tag{2.12}$$

h_i is the hidden state of the encoder and $a_{t,i}$ is the weight of this specific shortcut from the encoder to the decoder which is conditioned by the state of the decoder in time s_{t-1} and every single word h_i in the source sentence. In [Bahdanau et al., 2014], the scoring function $a(s_{t-1}, h_i)$ is realised as a feed-forward neural network.

$$\hat{a}_{t,i} = softmax(a(s_{t-1}, h_i))) \tag{2.13}$$

[6] This approach can be referred to as a bidirectional language model; however, they used a mechanism which does not allow cheating with using an inappropriate context.

[7] https://allennlp.org/elmo

Content-base attention	$a(s_t, h_i) = cos(s_t, h_i)$	[Graves et al., 2014]
Additive	$a(s_t, h_i) = v_a^T tan(W_a[s_t; h_i])$	[Bahdanau et al., 2014]
Location base	$a(s_t, h_i) = softmax(W_a s_t)$	[Luong et al., 2015]
General	$a(s_t, h_i) = s_t^T W_a h_i$	[Luong et al., 2015]
Dot-Product	$a(s_t, h_i) = s_t^T h_i$	[Luong et al., 2015]
Scaled Dot-Product	$a(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	[Vaswani et al., 2017]

Table 2.2: Overview of various $a()$ functions used in recent works.

In the original paper, the function a was realised as the so-called *additive attention*:

$$a(s_{t-1}, h_i) = v_a^T tan(W_a[s_{t-1}; h_i]). \tag{2.14}$$

After that, however, researchers started to explore plethora of other ways how to score a specific combination of s_t and h_i. An overview can be found in the Table 2.2.

In the *query-key-value* notation, the *key* is the state h_i of the encoder, the *query* is the state of the decoder s_t in time t and the *value* is the original hidden state of the encoder h_i. Fig. 2.11 depicts the standard encoder-decoder scheme with added attention.

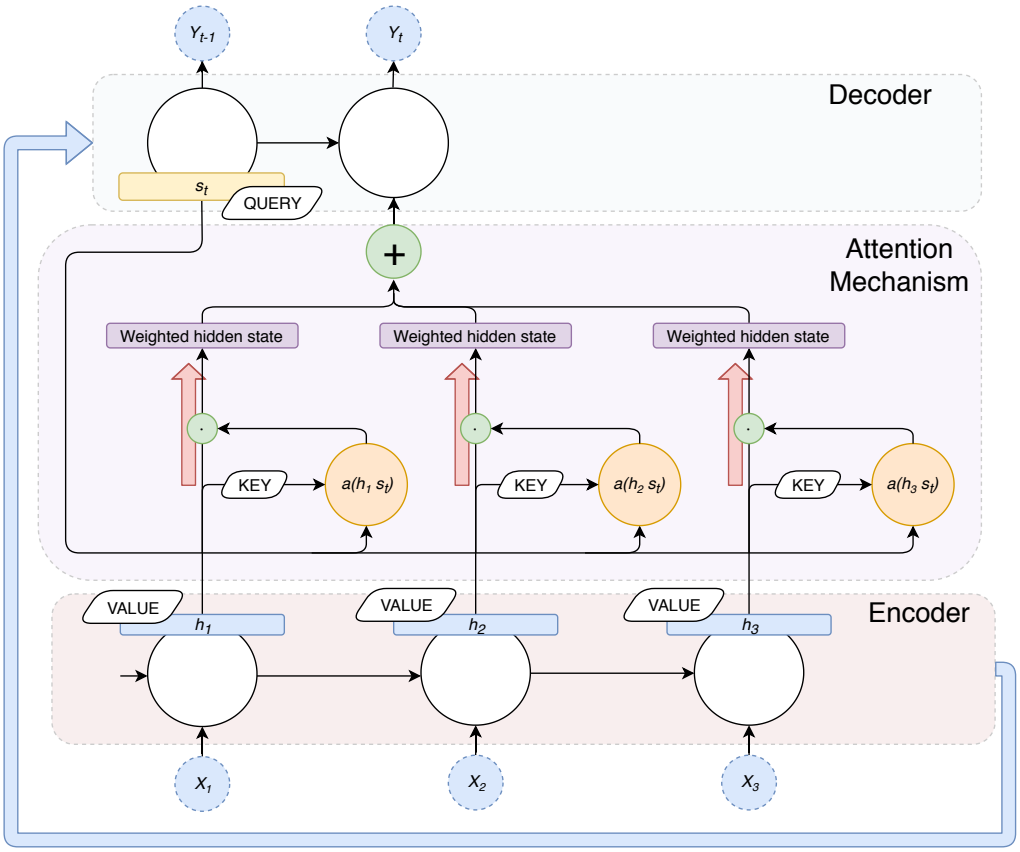


Figure 2.11: Visualisation of the attention mechanism. The query, key, and value correspond to Q K V notation from [Vaswani et al., 2017] with the Transformer model where the function a is realised as the dot product.

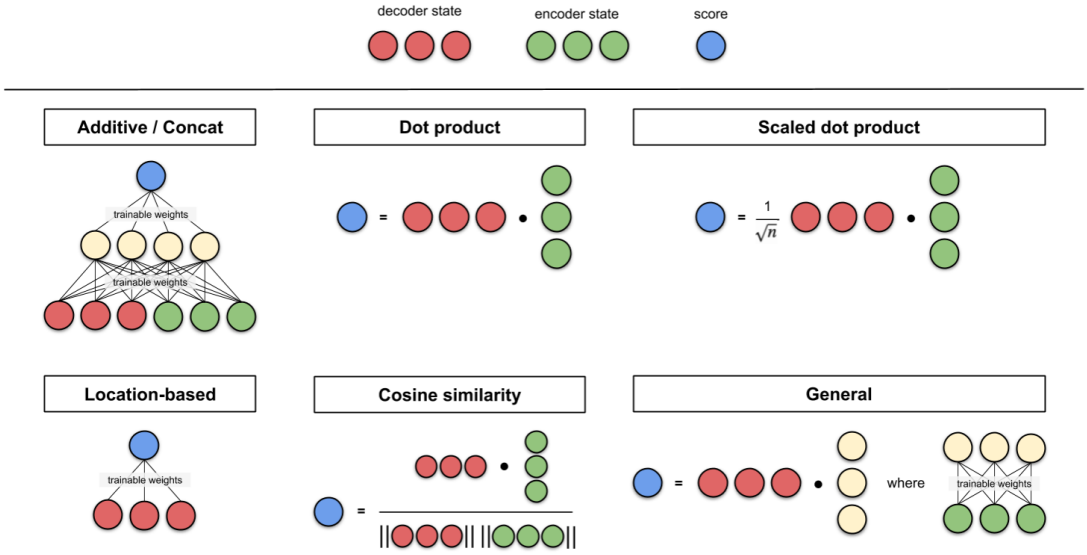


Figure 2.12: Visualisation of different attention mechanisms from Table 2.2. Figure was taken from [att, 2019].

2.5.1 Transformer

After some few papers presenting the high performance of the attention mechanism in different problems, the paper "Attention is all you need" was introduced by [Vaswani et al., 2017] in which a new model – the *Transformer* – was introduced.

The encoder of the transformer contains stacked *self-attention* layers. Self-attention (sometimes also called *intra-attention*) is the same mechanism as the standard attention introduced earlier, the only difference is that the conditioning vectors are not taken from the encoder to the decoder but they are used during the whole encoding process in each layer. In other words, the Transformer model is made up of multi-layered self-attention.

This relatively simple architecture brings many advantages. In text domain, robustness in the processing of the different lengths of its inputs is possibly the biggest one. However, the possibility of parallel calculations, no assumptions about the relationships across the data, or power to catch long-range dependencies are really remarkable.

Positional Encoding Nonetheless, if the input has some local or temporal relationship such as time series or text, some positional encoding must be added. Otherwise, the model will see a bag of words.

If the positional encoding is added, the information about the relative position of words in a sentence can be taken into account. The positional encoding vector is created from sines and cosines from the position in the sequence [Vaswani et al., 2017].

Another possibility is to use **positional embedding** where the precisely designed mechanism for position encoding is replaced by special positional embedding learned end-to-end with the rest of the model [Gehring et al., 2017].

Scaled Dot-Product Attention Because of the stacking of more attention layers, change of variance would be a problem. To avoid this problem, the dot-product attention is scaled by the factor $\sqrt{d_k}$ where d_k is a hidden dimension[8].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.15}$$

For example, consider that Q and K have a mean of 0 and variance of 1. Their matrix multiplication will have a mean of 0 and variance of d_k. Hence, the square root of d_k is used for scaling because the matrix multiplication of Q and K should have a mean of 0 and variance of 1, and we get better gradients for learning.

Why the variance of the dot products grows, is evident from the following equation of dot product if we consider q and k random variables with mean 0 and variance 1:

$$q \cdot k = \sum_{i=1}^{d_k} q_i k_i \tag{2.16}$$

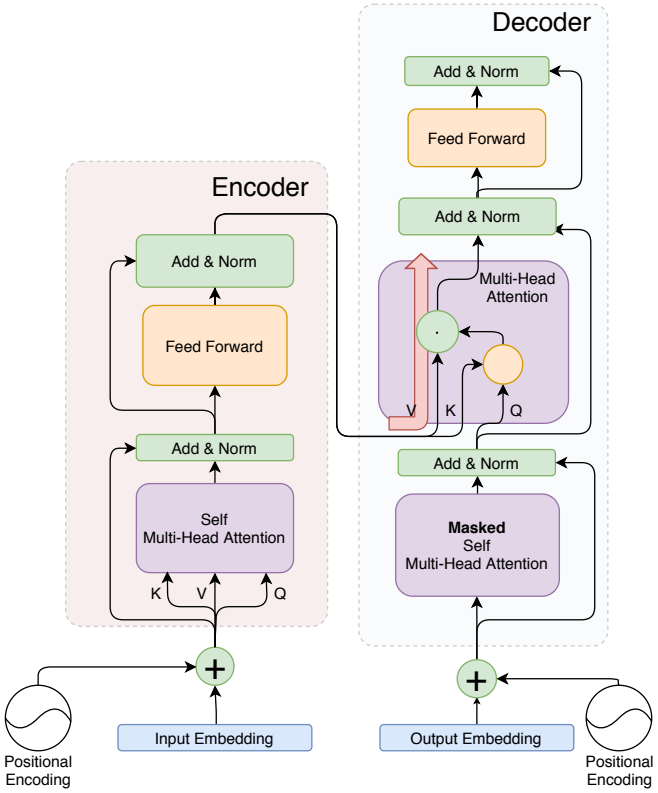


Figure 2.13: Transformer model scheme

The whole stack of the Transformer is depicted on Fig. 2.13.

[8]The dim(Q) and dim(K) are the same

2.5.2 Transformer in Action

After the Transformer introduction, lots of models were based on the same principle and brought significant improvements in lots of different domains.

USE – Universal Sentence Encoder Universal sentence encoder was introduced in [Cer et al., 2018]. The main idea is to create an universal sentence encoder on two different tasks, including language modelling task and SNLI[9] dataset (entailment, contradiction, and neutral). They explored two different models for sentences encoding: One based on the Transformer (sec. 2.5.1) and second based on Deep averaging networks[10]; they use a combination of both approaches for fine-tuning on final tasks.

Their final Transformer-encoder model makes a 512-dimensional vector from a lower-cased tokenised string as the sentence embedding. It uses the element-wise sum of word representations made up by the Transformer-encoder and divided by the square root of the length of the sentence – the short sentences are not dominated by the longer ones. The model is publicly accessible on *tf.hub*[11]

The encoder based on Deep averaging network [Iyyer et al., 2015] has the same input and output. The primary advantage of DAN is short computational time.

GPT – Generative Pre-Trained Transformer One of the most powerful architecture for training a sentence encoder is probably GPT [Radford et al., 2018]. The main advantage of this techinque pre-training of a high capacity language model on massive textual data in the first phase. Because of practically unlimited amount of this type of data, modern architectures can reach high capacity. In the second phase, this pre-trained language model is used for encoding text for fine-tuning on supervised tasks with labelled data. In this paper, they fine-tuned their models and measured performance on these tasks:

- natural language inference,
- question answering,
- sentence similarity,
- classification.

They use concatenation and use delimiter for tasks that have two sequences on the input. In their model, they used 12-layered decoder-only transformer with masked self-attention heads (768-dimensional states and 12 attention heads). For the position-wise feed-forward networks, they used 3072-dimensional inner state. Overall, it is 117 million parameters. The state-of-the-art model they used, two-phased training and multi-task fine-tuning with propagating errors into encoder make a robust model for many tasks using sentence encoding in any phase.

BERT – Bidirectional Encoder Representations from Transformers BERT was introduced in [Devlin et al., 2018], it is a model for making a word-in-context representation as well as representations of whole sequences of words. They were inspired by the creators of GPT in the representation of two sequences and the used delimiters; however, they added an embedded token type to support the information about the order of both sentences.

They introduced two models with different hyper-parameter settings. Their small model used L=12 Transformer blocks with H=768 dimensional states and A=12 self-attention heads, and feed-forward/filter size as 4H=3072. In total, 110 million parameters.

[9] The Stanford Natural Language Inference
[10] Word embeddings are summed up firstly and processed by multi-layered perceptron right after.
[11] https://tfhub.dev/google/universal-sentence-encoder/4

The large model used L=24 Transformer blocks with H=1024 dimensional states and A=16 self-attention heads, and feed-forward/filter size as 4H =4096. In total, 340 million parameters.

Both models are publicly available[12].

GPT-2 A year later, the same team introduced a similar model but based only on the first phase – language modelling [Radford et al., 2019]. Nonetheless, the dataset was massively extended up to 40GB of texts. The goal of experiments in this work was to measure how much it is possible to reach good performance in a wide range of tasks with the zero-shot setting (See Note 2 on page 16).

They present models in two different sizes:

- The smaller has 762 million parameters, and it is publicly accessible.
- The larger one has 1542 million parameters and its authors decided not to share it.

Note 2: Zero-Shot and Few-Shot Learning

The goal of *zero-shot* and *few-shot* learning is to solve standard complex NLP tasks like machine translation, question answering, Winograd schemas, common sense reasoning, reading comprehension with only language model training. There is no fine-tuning on end-tasks – the task is introduced in the form of natural language as a human would do.

In some tasks, they reached exciting results. Even if the product did not surpass the state-of-the-art unsupervised approaches, it brings a new idea of how to solve tasks with the zero-shot setting and shows a promising way.

GPT-3 The following model GPT-3 [Brown et al., 2020] showed impressive results with using high capacity language model in zero-shot and few-shot settings on various tasks.

However, the ratio of memorising and generalising remains still in question. A leak of test data into the training data set of such size[13] is hard to avoid.

In the paper, the authors also mentioned that such leak was confirmed in some setups. Despite the disclosure, the experiments were not repeated due to the size of the model (175 billions of parameters) and limited financial resources (approx. $4.6M). Nonetheless, these experiments showed an impressive capacity of the suchlike neural model. Even if all tasks were solved just by memorising and querying database built up this way, it is a phenomenal outcome.

Both the GPT-2 and GPT-3 used the language model in the *zero-shot* or *few-shot* setup, and there was no fine-tuning on end-tasks – the task was introduced in the form of natural language as a human would do.

[12] https://github.com/google-research/bert

[13] 45TB of compressed plain text before filtering and 570GB after filtering, roughly equivalent to 400 billion byte-pair-encoded tokens.

Chapter 3

Generative Models

In this chapter, we will get some fundamental knowledge about the differences among mainstream approaches and their similarities. Figure 3.1 depicts the taxonomy of generative models.

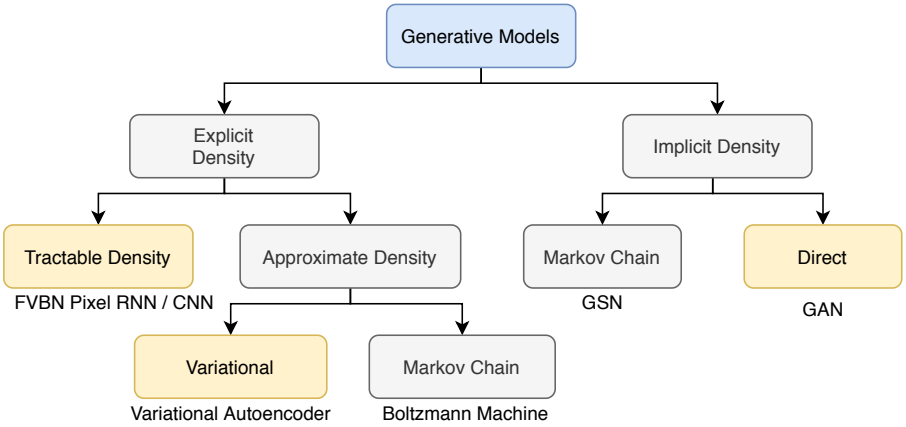


Figure 3.1: Generative models taxonomy, inspired by [Goodfellow, 2016]

It was shown that Deep Boltzmann Machine significantly outperforms SVMs and LDA on some discriminative tasks [Srivastava and Salakhutdinov, 2012]. However, they are not usually used nowadays, probably due to the Markov chain which prevents the model from scaling well. Therefore, this approach will not be discussed in details in this work.

GANs were designed to address big issues in the known models – Markov chains often fail to scale to high dimensional spaces and computational costs are higher. That leads to unpopularity of the generative model (GSN) use. Variational approximation of density often leads to *blurry* samples due to modelling the variation of samples; generators created from VAE (variational autoenoder) tent to generate *average* samples. However, it was shown that VAE architecture is used in some applications; approaches combining VAE and GAN are also known.

In the group of models with tractable density, there dominate *Fully visible belief networks* – FVBN (see Sec. 3.1.1 on pg. 18).

FVBN, VAE, and GAN can be presented as today's most used approaches in generative tasks – they will be discussed on the following pages.

Before we dive into models using deep learning, we should mention that although deep learning

approaches are dominant in some tasks, and we know modern approaches like adversarial training, there are still problems that are much more straightforward to solve in a canonical way. A typical example of such a problem can be the Electra model in which *MLE* (maximum likelihood estimation) dominates over the adversarial approach (see Sec. 3.5.4 on pg. 36).

3.1 Explicit Models with Tractable Density

The main feature of this type of models is that they return explicit probability densities unlike their alternatives such as generative adversarial networks (see Sec. 3.3.1 on pg. 22). It can be useful in some applications that need probabilistic planning and exploration of their environment [Bellemare et al., 2016].

There are two main approaches to make such models: **Fully Visible Belief Networks** and **Nonlinear Independent Components Analysis**.

3.1.1 Fully Visible Belief Networks – FVBN

This type of model uses the chain rule of probability to decompose the generative process of n-dimensional vector into a product of one-dimensional probability distributions [Frey et al., 1996].

$$P(x) = \prod_{i=0}^{n} P(x_i | x_0, ..., x_{i-1}) \tag{3.1}$$

The straightforward way to generate samples similar to those from some data set is modelling them by conditional probability of elementary entities defining a specific sample.

This method is quite often used in contemporary applications. In image generation, it could be prior pixels in some sequence defined by some order[Oord et al., 2016b]. In the text generation, previously generated characters, sub-words or words can be used for conditioning the probability distribution over the vocabulary, and by sampling, we can emit the next words.

The main disadvantage is non-parallelizability – each sample needs to be generated sequentially, i.e. atom by atom. This approach is one of the most popular nowadays and models like PixelRNN and PixelCNN [Oord et al., 2016b], WaveNet [Oord et al., 2016a] produce exciting results. However, they are often slow and using them in a real application is not possible at this moment – WaveNet needs about two minutes of computations to generate one second of audio stream. Nonetheless, there are experiments that modified models which scale better [Paine et al., 2016, Oord et al., 2017].

Conditioning Adding new conditions on a higher level is straightforward. However, having labelled training data is a hard-to-meet condition. With this approach, we can add any type of additional labels to the process which prepares conditional probabilities in any suitable form (e.g. one-hot, embedding). The capacity of such a system depends on complexity of the added information, the capability of the used model and size of the data set. The one-hot encoding that specifies a class is equivalent to adding a class-dependent bias at every time step. We can also use this approach to force the generator to generate high-level features at a specific time of the generation. For example, we could specify that a animal object should appear in the generated image but may do so in different positions and poses and with different backgrounds.

3.1.2 Nonlinear Independent Components Analysis

Nonlinear ICA (Independent Components Analysis) is also one of the methods with explicit density functions [Hyvärinen and Pajunen, 1999]. The goal is to find nonlinear transformation $g(z)$ which maps variable z from a latent space onto a real data space. The idea is trivial; however, there are lots of restrictions for such function $g(z)$. This transformation must be continuos, differentiable and invertible. It means that the latent space and the real data space must have the same dimensionality what is inconvenient and impractical. This is probably the main reason why this approach is not usually used in recent works [Dinh et al., 2016]. The first model of this kind was introduced in 1995 by Deco and Bauer [Deco and Brauer, 1995].

3.2 Explicit Models with Approximate Density

To extend the models and the possibility of adding some uncertainty, methods modelling some kind of approximation were invented. There are two main categories:

- deterministic approximations – variational methods,
- stochastic approximations – Markov chain Monte Carlo methods.

3.2.1 Variational Autoencoder – VAE

This approach models intractable density function but uses a tractable density approximation. Intractable density is here because of the need of marginalising out over the random variable z (Eq. 3.2).

We are forced to use a variational approximation instead – distribution $Q(z)$ which lower-bounds the true density. These models are often good at obtaining a high likelihood but they tend to produce "blurry" samples.

Variational Autoencoder is recently the most popular deep model inspired by the standard autoencoder scheme [Hinton and Salakhutdinov, 2006]. The first part of the autoencoder model – the encoder – tries to reduce the dimensionality of the data into a single vector z preserving as much information as possible. The second part of the model – the decoder – tries to reconstruct the original data from this vector (Figure 3.2). A trained model may be used possible to use for data dimensionality reduction.

Note 3: Latent Variable Model

Latent variable model is a statistical model that describes relationship between a set of latent variables and a set of observable variables (sometimes called manifest variables). This model is based on dependence of each observable variable on a combination of the latent variables. In this sense, we can interpret the observed sample as a projection of a less-dimensional latent variable with ideally independent components onto a high-dimensional manifest variable with dependent components. Some of the currently most used models are based on this principle; the variational autoencoder is directly derived from this scheme (see Sec. 3.2.1 on pg. 19).
The following equation describes the mathematical model.

$$P(X) = \int P(X|z;\theta)P(z)dz, \tag{3.2}$$

where θ represents the parameters of the model and z is a random variable.

VAE is a generative model based on the latent variable model (See Note 3 on page 19). After the

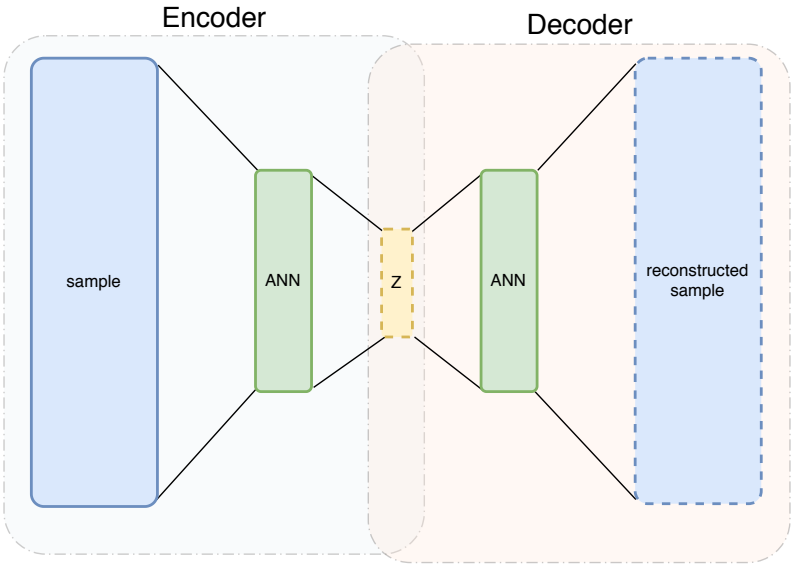


Figure 3.2: Visualisation of the autoencoder scheme

training, it is possible to sample the random latent variable z from $\mathcal{N}(\mu, \sigma^2)$ and use the decoder to generate a new sample.

The latent space is sampled for generating similar samples to the real data and it is typically less-dimensional; therefore, it should catch latent information in a form which is more abstract. This space can be non-intuitive for humans though; however, there are methods to deal with it for a specific application (See pg. 21 – Disentangled VAE).

Before we can say that our model is representative for our data, we need to make sure that for each data point X we can find one or more settings of the latent variable z which causes the model to generate a sample very similar to X

Objective Variational autoencoder uses a specific way to compute Equation 3.2. There is a shortcut we can take[1]. For a lot of zs the $P(X|z)$ will be nearly zero. The main idea is to take only zs that led to producing a good X and compute $P(X)$ just from those. Thus, we need a new function $Q(z|X)$ which can give us from X a distribution over zs that are likely to produce X. The space of z values that are likely under Q will be much smaller than the space of all zs that are likely under the $P(z)$. This enables us to compute $\mathbb{E}_{z\sim Q}P(X|z)$ relatively easily.

The relation between $\mathbb{E}_{z\sim Q}P(X|z)$ and $P(X)$ is one of the cornerstones of variational Bayesian methods. At first, we define Kullback-Leiber divergence (signed as KL or $\mathcal{D}$) between $Q(z)$ and $P(z|X)$.

$$\mathcal{D}[P(z|X)||Q(z)] = \mathbb{E}_{z\sim Q}[\log Q(z) - \log P(z|X)] \tag{3.3}$$

By applying Bayes rule to $P(z|X)$, we get both $P(X)$ and $P(X|z)$ into the equation:

$$\mathcal{D}[P(z|X)||Q(z)] = \mathbb{E}_{z\sim Q}[\log Q(z) - \log(X|z) - \log P(z)] + \log P(X). \tag{3.4}$$

Here, $\log P(X)$ can be separated from the expectation because it does not depend on z. Negating both sides, rearranging and contracting par of $\mathbb{E}_{z\sim Q}$ into KL-divergence terms yields:

$$\log P(X) - \mathcal{D}[Q(z)||P(z|X)] = \mathbb{E}_{z\sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)||P(z)] \tag{3.5}$$

[1]This part was inspired by [Doersch, 2016]

This equation is the core of variational autoencoder. In short, left-hand side expresses the quantity we want to maximise: $\log P(X)$ plus an error term makes Q produce zs that can reproduce a given X – this term becomes small if Q has a high capacity. The right-hand side is something we can optimise via stochastic gradient descent. Note that the right-hand side looks like an autoencoder in which Q is encoding X into latent zs and P is decoding zs to reconstruct X.

If we want to perform gradient descent on the right-hand side of Equation 3.8, we have to specify a form of $Q(z|X)$. The usual choice is set $Q(z|X) = \mathcal{N}(z|\mu(X;\theta), \Sigma(X;\theta))$, where μ and Σ are deterministic functions with parameters θ that can be learned from data. In practice, μ and Σ are again implemented as neural networks.

The last term of Equation 3.8 – $\mathcal{D}[Q(z|X)||P(z)]$ – is the KL-divergence between two multivariate Gaussian distributions which can be computed in closed form as:

$$\mathcal{D}[\mathcal{N}(\mu_0, \Sigma_0)||\mathcal{N}(\mu_1, \Sigma_1)] = \frac{1}{2}\left(tr\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^T \Sigma_1^{-}1\,(\mu_1 - \mu_0) - k + \log\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right) \tag{3.6}$$

where k is the dimensionality of the distribution. It can be simplified in our case to:

$$\mathcal{D}[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)] = \frac{1}{2}\left(tr(\Sigma_1(X)) + (\mu(X))^T(\mu(X) - k + \log\left(\det\Sigma\left(X\right)\right)\right) \tag{3.7}$$

In the first term on the right-hand side of Equation 3.8, we could use sampling of z as well as sampling of X from our data to estimate $E_{z\sim Q}[\log P(X|z)]$.

The full equation is:

$$\mathbb{E}_{X\sim D}\left[\log P\left(X\right) - \mathcal{D}\left[Q(z)||P(z|X)\right]\right] = \mathbb{E}_{X\sim D}[\mathbb{E}_{z\sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)||P(z)]] \tag{3.8}$$

For gradient descent optimisation, we can sample a single value of X and a single value of z from the distribution $Q(z|X)$ and compute the gradient of:

$$\log P(X|z) - \mathcal{D}[Q(z|X)||P(z)] \tag{3.9}$$

Reparametrisation In order to get the whole system learnt by gradient descent, it is impossible to have the sampling operation in the process. A reparametrisation trick solves this issue by pulling parameters of the distribution from the stack. Those parameters are set as constant values and the variable is scaled with respect to mean and variance after the sampling operation (Figure 3.3).

Disentangled VAE Several research teams are investigating the possibility of disentanglement of the VAE latent space. Disentanglement in this context means forcing the model to use human-understandable bases of the latent space which leads to a possibility to apply predictable changes in the output of VAE thus making changes in the latent variable z. It can mean forcing a generator to use a specific style in hand-writing; changing sex, age, hair colour or even position of the camera in a generated face picture [Bouchacourt et al., 2018]. However, it is not necessary to disentangle latent space in many cases. It can be useful for expanding data sets in semi-supervised learning [Li et al., 2019, Hsieh et al., 2018].

3.2.2 Deep Boltzmann Machine

Deep Boltzmann Machine [Salakhutdinov and Hinton, 2009] is based upon Restricted Boltzmann Machine (RBM) [Hinton and Salakhutdinov, 2006] belonging to the group of autoencoders and

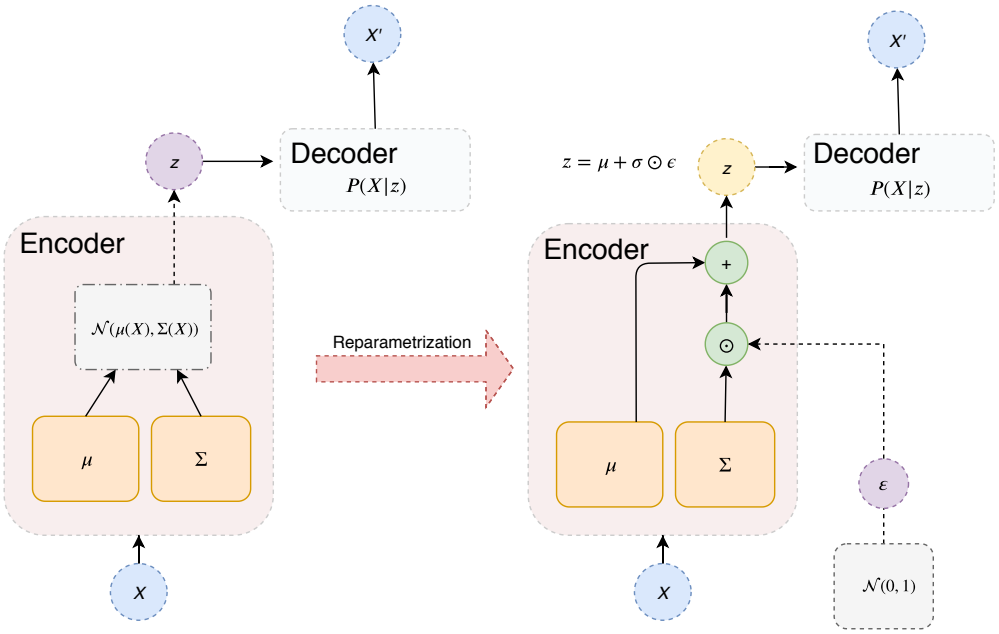


Figure 3.3: Reparametrisation trick

can be used for extracting essential features as well as a generator. It is based upon the Boltzmann Machine; the restriction is here in the assumption of independence of units on the same level.

This simple idea is used in modern models for building larger models called Deep Boltzmann Machines (DBM) or sometimes also called Deep Belief Networks (DBN). They help to deal with some significant issues in deep learning like the vanishing gradient or the lack of labelled data (see Sec. 4 on pg. 45) [Zhou et al., 2010]. As any autoencoder, the Boltzman machine or some special deeper variant can be used as a generator by sampling from the latent space. However, there is still a problem with the underlying Markov chain which do not scale well for larger problems.

3.3 Implicit Density Models

3.3.1 GAN – Generative Adversarial Networks

Generative adversarial approach is based on the Generator-Discriminator model (Figure 3.4).

The goal of the generator is to generate samples that are indistinguishable from training data for the discriminator. In game theory, this kind of conflict of interests is known as the so-called zero-sum game.

$$l_G(\theta_D, \theta_G) = -l_D(\theta_D, \theta_G) \tag{3.10}$$

Training of an adversarial model consists of finding Nash equilibrium of a two-player non-cooperative game.

Generator The goal of the generator is to generate samples of some latent variable z and to provide samples that are similar to real data. Variable z is randomly taken from a latent variable space and can be seen as some seed into the generation process. Along with the change of the z value, different samples similar to the real data should be generated. The variability of samples

coming from the generator is a serious topic and it is one of the most investigated behaviour of the generator (see Mode Collapse on page 30).

When we speak about GAN (Generative adversarial network), we mean the specific adversarial model which consists of two neural networks: One is in the role of the generator and the second one in the position of the discriminator.

The generator tries to generate samples from *random* noise that are as close as possible to the real data distribution, and in the vanilla GAN, the generator has no access to any real data – it is trained only on gradients propagated through the discriminator.

However, the generator can be implemented as naïve bayes, mixtures of multinomials, mixtures of gaussians, HMM etc; although it is not usual nowadays.

Discriminator The goal of the discriminator is to distinguish between real and fake examples. Its role is important because gradients to the generator are provided only from the discriminator. Therefore, if the discriminator acts randomly, there is no usable information to update the generator.

Discriminator can be implemented as any kind of classifier, e.g. Logistic Regression, Support Vector Machine, Neural Network, Nearest Neighbour, Conditional Random Field...

[Goodfellow et al., 2014] showed that, with respect to an optimal discriminator, the minimax formulation[2] can be shown to minimise the Jensen Shannon Divergence between the generator output distribution and the real data distribution.

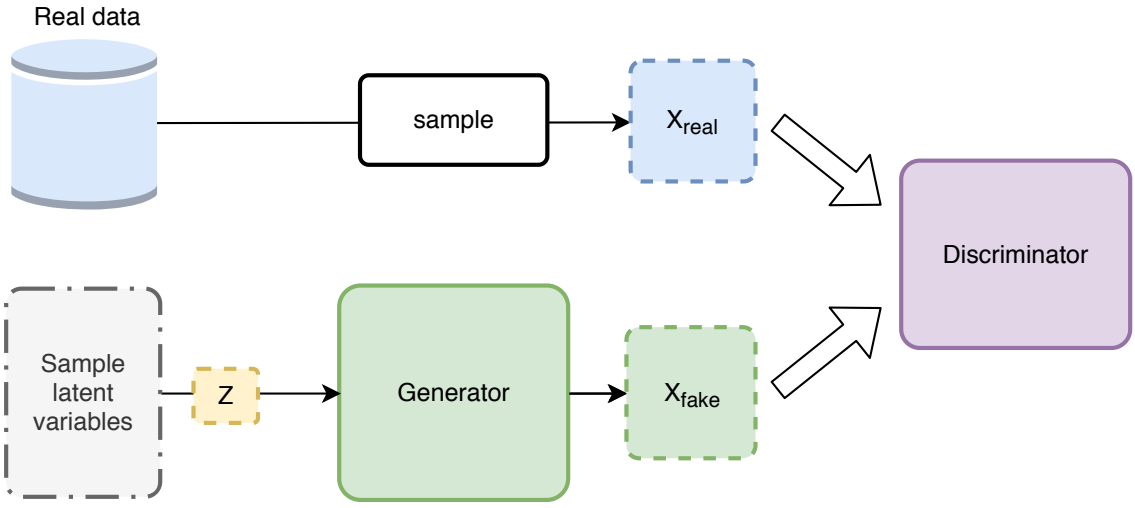


Figure 3.4: Model Generator-Discriminator

In simple terms, the first part of the network generates random samples from some distribution and the second part tries to recognise whether the samples are from real data or were produced by the generator. The loss function can be formalised as follows:

$$l_D(\theta_D, \theta_G) = -\frac{1}{2}\mathbb{E}_{x\sim P_{data}} logD(x) - \frac{1}{2}\mathbb{E}_z log(1 - D(G(z)) \tag{3.11}$$

[2]The discriminator maximises the error of distinguishing between real and fake data, while discriminator minimises the same.

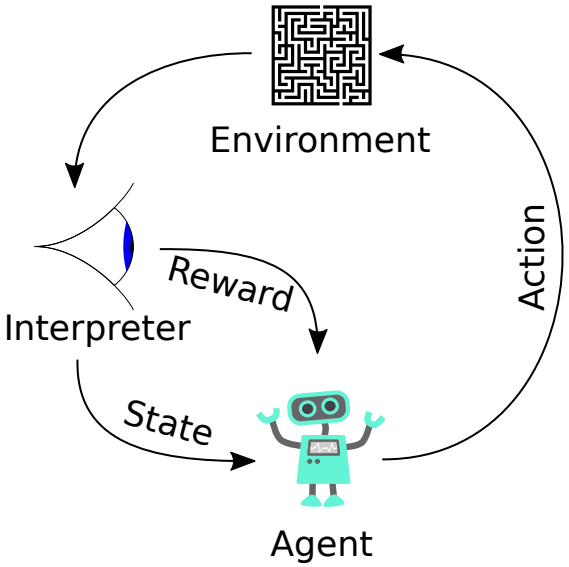


Figure 3.5: Typical paradigm of reinforcement learning. (image from https://commons.wikimedia.org/)

D indicates the discriminator and G the generator. The term $D(G(z))$ is the answer of the discriminator to the sample generated by the generator from the sampled latent variable z. The $D(x)$ is the answer of the discriminator to the real data sample x.

3.3.2 GANs on Text

Because of the need for propagating errors between the generator and the discriminator, a hard discrete stochastic decision like word-sampling is an issue. A text is usually generated by a hard decision made after the softmax layer over the token space.

There are few principal approaches to deal with the problematic discrete stochastic unit in this setup:

1. Approaches using reinforcement learning methods (SeqGan [Yu et al., 2017]).
2. Reparametrization using Gumbel-Softmax (also known as the Concrete distribution) ([Kusner and Hernández-Lobato, 2016, Jang et al., 2016]).
3. Staying in continuous space [Makhzani et al., 2015, Donahue and Rumshisky, 2018].

Another completely different option is not to have a discrete stochastic unit at all on the output of the generator (e.g. generating tokens deterministically). That would eliminate the original problem of backpropagating gradient from the discriminator to the generator.

3.3.3 Reinforcement Learning – RL

Markov Decision Process Reinforcement learning is based on agent-like approach. The agent can interact with an environment to reach its goal.

In the special case – the text generation task – in which the agent is the generator, the emitted word is the taken action that may bring some reward due to the fact that the discriminator can

not distinguish between the real and fake sample. Every word we have already written defines the sequence states we went through. In every natural language, this leads to combinatoric explosion because of the number of words in the vocabulary. Because of this, choosing the right word from tens of thousands of all possible choices in every time step is a really hard problem. Moreover, no reward will come until the generative process ends up.

We also have one big problem in RL. The reward could potentially come much later after taking the action. Between choosing the action and the final state in which we get a reward, there could be lots of other decisions which potentially do not contribute at all. To learn which decision to make now – which word to write – when we do not have any reward until the discriminator gets the whole sentence, is a specific and not an easy problem.

RL algorithms can be compartmentalised upon several attributes.

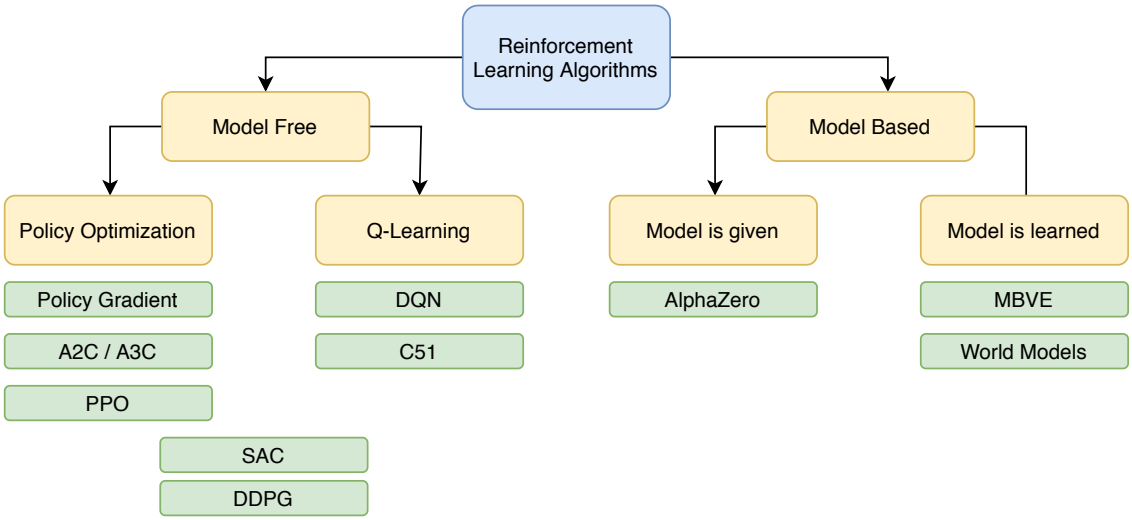


Figure 3.6: Reinforcement learning taxonomy [Achiam, 2018]

Access to the model If the agent has access to the model of the environment, we speak about Model-Based RL. The main advantage is that it allows the agent to plan by considering the future via going through possible choices (See 3.3.6). However, we usually do not have a ground-truth model of the environment. Therefore some approximation is often used. Other methods can be referred to as *Model-Free RL*.

Optimisation If the policy of the agent is directly optimised, the term *Policy Optimisation* is used. Asynchronous Methods (A2C/A3C) [Mnih et al., 2016] and Proximal Policy Optimisation [Schulman et al., 2017] belong to this group. *Q-Learning* [Watkins, 1989] uses a little different approach. The goal is to learn an approximator $Q_\theta(s, a)$ which should estimate maximal possible reward of the state-action pair. Because the Q-learning is an off-policy method, we can use the data collected at any point during the training and it is independent of how the agent was acting to get this data. The main representatives of this group are DQN [Mnih et al., 2013] and C51 [Bellemare et al., 2017].

Several works are dealing with combining Policy Optimisation and Q-Learning and trying to find some trade-off to benefit from positive aspects of each one.

On-policy vs. Off-policy The methods can be also split into two groups according to order whether they use policy-dependent (*on-policy*) or policy-independent (off-policy) data.

3.3.4 Value Function

In order to train an agent for using an optimal policy, it is crucial to have some way to derive the value of a state or an action-state pair. The basic *On-Policy Value Function* gives the expected reward returned in the state s according to the fixed policy used in every future step. It can be formalised as:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[R(\tau)|s_0 = s]. \tag{3.12}$$

The $R(\tau)$ is the reward for the trajectory τ, π is the policy with parameters θ and the s_0 is the initial state.
The version extended with the action – **On-Policy Action-Value Function**– the so-called Q function which gives the expected reward in the state s taking action a:

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[R(\tau)|s_0 = s, a_0 = a]. \tag{3.13}$$

The optimal value function V^* and the optimal action-value function Q use the assumption that at any time we choose the best action in order to maximise the reward. In other words, if we used optimal policy, the on-policy value function should be the same as the optimal one.

$$V^*(s) = \max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[R(\tau)|s_0 = s] \tag{3.14}$$

$$Q^*(s, a) = \max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[R(\tau)|s_0 = s, a_0 = a] \tag{3.15}$$

The optimal action a^* selected by the optimal policy is the action which maximises optimal action-value function $Q^*(s, a)$,

$$a^*(s) = \arg\max_{a} Q^*(s, a). \tag{3.16}$$

We can use Bellman's principle of optimality, which is customary and useful for the process of learning, and form all four equations as follows:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\substack{a \sim \pi \\ s' \sim P}} \big[R(s, a) + \gamma V^{\pi}(s')\big], \tag{3.17}$$

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{s' \sim P} \big[R(s, a) + \gamma \mathop{\mathbb{E}}_{a' \sim \pi}[Q^{\pi}(s', a')]\big], \tag{3.18}$$

$$V^*(s) = \max_{a} \mathop{\mathbb{E}}_{s' \sim P} \big[R(s, a) + \gamma V^*(s')\big], \tag{3.19}$$

$$Q^*(s, a) = \mathop{\mathbb{E}}_{s' \sim P} \big[R(s, a) + \gamma \max_{a'} Q^{\pi}(s', a')\big]. \tag{3.20}$$

s' indicates that the next state is sampled from the environment transition rules using action a in state s. $\gamma \in (0, 1)$ is a discount factor which prioritise earlier rewards instead of later ones.

3.3.5 Policy Gradient Method

Policy gradient method, sometimes also called Monte Carlo Policy Gradients (in combination with MCTS – see 3.3.6), is a method for handling large space of possible actions. The policy specifies what action should the agent take in every specific state of the process – which word to use in text generation. Policy gradient methods learn the policy as a classification problem on top of some features extracted from the actual state of the environment.

The objective is to maximise a cumulative reward to be received in any state of the system using available actions. If we consider generating words as taking actions, we have a discrete and finite action space.

In the reinforcement learning, the term *trajectory* is commonly used for sequence of states of the environment.

Probability of a trajectory from a starting state to a final state using policy π_θ is:

$$P(\tau|\theta) = p(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t). \tag{3.21}$$

The objective can be formalised as

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathbb{E}[R(\tau)] && (3.22)\\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) && (3.23)\\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) && (3.24)\\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) && (3.25)\\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && (3.26)\\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)] && (3.27)
\end{aligned}$$

We can estimate gradients by sampling trajectories by letting the agent act in the environment using the policy.

$$\nabla_\theta J(\theta) \sim \frac{1}{D} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau) \tag{3.28}$$

r_{t+1} is the reward received by taking the action a_t at the state s_t. This is a basic form of policy gradient and if we are able to use policy π_θ in the environment, we can collect some trajectories and update our policy to optimise getting a reward.

The function R (see Sec. 3.3.9 on pg. 29) models the reward received by the agent and the appropriate setup of such function is the key to a stable training process.

3.3.6 Monte Carlo Tree Search – MCTS

Using the Monte Carlo search when expanding a state tree is a straightforward process. Instead of expanding the whole sub-tree, which is inefficient or even impossible in some cases, we can sample from actions used for expansions according to some heuristic.

For guiding Monte Carlo search to expand a tree better and avoid poorly rewarded states, [Kocsis and Szepesvári, 2006] developed a method which prioritise better sub-tree to be expanded – UCT (upper confidence bounds applied to trees).

After having used it in the MoGo program, [Gelly et al., 2006] reached impressive results in playing 9x9 Go with human players.

However, the UCT faces criticism for its over-optimistic assumptions [Coquelin and Munos, 2007].

In the text domain, Monte Carlo is used for rolling out the rest of the sequence – taking ten best possible solutions and using the mean of them [Yu et al., 2017].

The solution with the MCTS showed its qualities on different tasks. However, searching the appropriate heuristic for choosing the right action to take is not an easy task.

3.3.7 Deep Q-Learning – DQL

The first try of using an approximation function for the expected reward (the so-called Q-function) by tree search while using neural network was introduced on the game of backgammon [Tesauro, 1994]. While playing with itself (the so-called self-play), the agent achieved super-human performance. It was model-free reinforcement learning similar to the one which laid the foundation for today well-known Q-learning. For the approximation of the Q value, the multi-layer perceptron with one hidden layer was used.

3.3.8 Experience Buffer for Off-policy Methods

In order to avoid oscillations, local extreme sticking and to increase data efficiency, the experience buffer was developed to smooth experiences over a longer time period [Mnih et al., 2013]. Instead of updating policy in every step, we just execute an action in the environment, observe the reward and store $s_t; a_t; R_{t+1}; s_{t+1}$ for later reuse.

Because of nonexistence of true data for minimising the error of our estimator,

$$\sum (Q(s, a - Q^*(s, a))^2, \tag{3.29}$$

the true value is considered as:

$$Q(s, a) = R_{t+1} + \gamma max_{a'} Q^*(s', a')), \tag{3.30}$$

where γ is discount factor.

Instead of directly using a reward from the environment, we use a Q-function which approximate the expected reward. It brings better data efficiency and avert divergence of the model.

Q-learning methods can reuse data more effectively than policy optimisation methods. However, they are also prone to failure and are less stable [Tsitsiklis and Van Roy, 1997].

3.3.9 Actor-Critic Notation

In modern methods, the Actor-Critic notation it is often used where the objective equation is composed of two terms.

- The *Actor* part updates the policy distribution in the direction suggested by the Critic (as with policy gradients).

- The *Critic* part estimates the value function. This could be the action-value function (the Q value) or the state-value (the V value).

We can derive the Actor-Critic formula from standard policy gradient as follows:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)] \tag{3.31}$$

By the definition of the Q-function (action-value function), the reward expected to be achieved for trajectory τ,

$$\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[R(\tau)] = Q(s_t, a_t), \tag{3.32}$$

can be modified by replacing the term $R(\tau)$ by $Q(s_t, a_t)$.

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \underbrace{\log \pi_\theta(a_t|s_t)}_{\text{Actor}} \underbrace{Q(s_t, a_t)}_{\text{Critic}}] \tag{3.33}$$

With this form, we can assign specific parts roles of the actor and the critic.

The Actor-Critic scheme is used in revolutionary AlphaGoZero [Silver et al., 2017]. The actor and the critic share a part of the neural network according to the fact that the raw playing field input should be extracted into some more abstract game state upon which the output of the actor and the critic could be made.

Reward shaping

Designing the reward function by a human can be tricky. There are well-documented cases, when the agent learned to hack the reward and reached very high reward score while never solved the basic problem at all. Because of the absence of knowledge about the real world, the reward can be designed in a inconvenient way and may be misinterpreted by the agent who is not biased by conventional practices.

3.3.10 Gumbel-Softmax Reparametrization

Another way of dealing with the discrete choice according to the need of differentiable operation in generative models is to use some approximating distribution. [Kusner and Hernández-Lobato, 2016] used Gumbel-Softmax (Eq. 3.34) function to deal with this problem. The Gumbel distribution is used to model a probability that some sample is maximum of the samples taken from the same distribution. Instead of using argmax for creating a sharp decision to choose one word to be generated, we can estimate directly the probability of being maximal. It helps us to solve the problem of indifferentiable decision between the generator and the discriminator.

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \tag{3.34}$$

The reparametrisation trick (see Sec. 3.2.1 on pg. 21) is used here and g_i is sampled from Gumbel distribution. π is the class probability, τ is the temperature added as a parameter changing during the training phase from a certain value to zero at the end. This causes relaxation at the start of the training (it is uniform as τ approaches ∞). As the softmax temperature τ approaches 0, samples from the Gumbel-Softmax distribution become one-hot, and the distribution are becoming identical to the original categorical distribution.

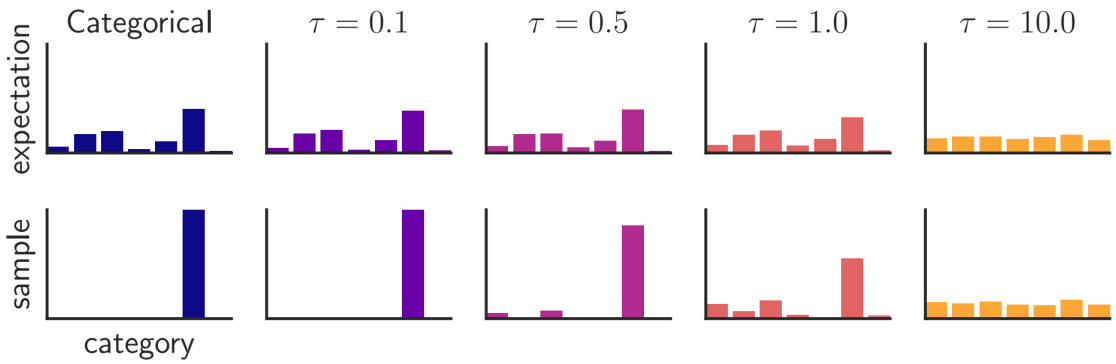


Figure 3.7: Gumbel-Softmax interpolates between discrete one-hot-encoded categorical distributions and continuous, categorical densities. Image was taken from [Kusner and Hernández-Lobato, 2016].

3.3.11 Problems with Adversarial Training

In generative adversarial networks, specific problems can show up during training due to the adversarial training mechanism in addition to those that are common for neural networks generally. A list of the most discussed ones follows with a study of possible solutions.

Mode Collapse The Mode Collapse problem leads to a poor output of the generator. It learns always to emit the same or a very small set of examples, and a wide scale of diverse latent vectors are projected onto the same output [Salimans et al., 2016].

This problem is one of the most common in the GAN architectures. How much this scenario influences the final application depends on the task. In some cases, we can be satisfied if the output is encapsulates the right information (in image from caption generation task) [Reed et al., 2016], or the variability of the output samples is important (random face generation, storytelling, etc.).

Of course, we also have to mention the role of conditioning in the mode collapse problem. It is intuitive that with every new condition, we force the generator to produce more bounded samples; this possibly leads to a space too narrow to allow for a reasonable diversity of the samples. If a training set is biased at the class level, we can also expect low diversity of generated samples; however, this emergent behaviour can be hidden much deeper in the data set and may arise only with complicated conditioning.

Lack of Gradient In a specific case, the generator can suffer from a lack of gradient. This situation will occur when the discriminator dominates and is successful in predicting the right class for each sample. There are some approaches for avoiding this symptom of pathological behaviour such as one-side label smoothing or non-saturating game (see Sec. 3.3.12 on pg. 31).

Stable Orbit As there are two players, we have two different objective functions. The convergence process can get stuck if we minimise the cost of each player simultaneously; a modification of $\theta^{(D)}$ that reduces $J^{(D)}$ can increase $J^{(G)}$ and vice versa and gradient descent enters an orbit, see Figure 3.11a [Salimans et al., 2016].

Adversarial Examples Due to the principle of the adversarial training, theoretically, the game of two players may drift away from the real world. The generator could learn to generate examples to fool the discriminator; however, that does not mean they are an acceptable result for humans. Some studies were conducted to research this topic. [Liu et al., 2016] showed on a black-box commercial classification system it is possible to design a sample precisely, so that it is 100% classified as a panda bear but a human would say with high confidence it is a banana. The term *Adversarial Example* has been established and attacking the neural network this way have become a serious topic. Theoretically, it is possible that the generator learns to fool the discriminator with adversarial examples. This would mean positively looking metrics during the training but incomprehensible results for humans. More about adversarial examples is written in sec. 3.5.8.

3.3.12 Methods for Learning Improvement

Discriminator and Generator Pretraining [Yang et al., 2017] show that pre-training of both the discriminator and the generator has a significant impact on the stability of the training. They tried to pre-train both to various values of accuracy. Initial accuracy of the discriminator needs to be set carefully and once it is too high (0.9 or 0.95) or too low (0.6 or 0.7), the model performs poorly. This suggests it is important for the generator and the discriminator to keep a balanced relationship at the beginning of the adversarial training. If the discriminator is too strong, the generator is always penalised for its wrong predictions and gets no idea about the right predictions. Hence, the generator is discouraged all the time and the performance can not be improved by giving poor information about the right gradient. On the other hand, if the discriminator is too weak, it is unable to give right guidance to the generator, i.e. the gradient direction used for updating the generator is random, too.

WGAN Another approach is to use a different metric to compute the distance between the distribution of the real data and the generated ones. Standard distribution similarity metrics equal zero if distributions are absolutely different which is a problem for gradient descent methods. [Arjovsky et al., 2017] suggest to use Earth-Mover distance (EMD) to improve model convergence. EMD is informally defined as the minimum cost of transporting mass in order to transform the distribution $\mathcal{Q}$ into the distribution $\mathcal{P}$ (where the cost equals to the mass times the transport distance). We assume distributions $\mathcal{Q}$ and $\mathcal{P}$ to be split into m and n clusters,

$$\min_{F} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} d_{i,j}, \tag{3.35}$$

$$EMD(\mathcal{P}, \mathcal{Q}) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} d_{i,j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j}}, \tag{3.36}$$

where $f_{i,j}$ is the flow from i to j, and $d_{i,j}$ is the distance.

EMD can be formulated and solved as a transportation problem using any algorithm for minimum cost flow problem, e.g. network simplex algorithm.

Non-saturating Game Due to the zero-sum game and the gradient descent optimization of arguments of the generator and the discriminator, the system is prone to non-convergent behaviour, e.g. to the stable orbit scenario.

To avoid this problem, the non-saturating game was introduced. Instead of using the negative loss of the discriminator for the generator (flipping the sign),

$$J^D = -\frac{1}{2}\mathbb{E}_{x\sim P_{data}} logD(x) - \frac{1}{2}\mathbb{E}_z log(1 - D(G(z)), \tag{3.37}$$

$$J^G = -J^D, \tag{3.38}$$

the cross-entropy minimization is suggested for the generator.

$$J^G = -\frac{1}{2}E_z \log D(G(z)) \tag{3.39}$$

Both are monotonically decreasing in the same direction. However, experiments have proof better convergence of non-saturating game [Goodfellow, 2016].

One-Sided Label Smoothing [Salimans et al., 2016] suggested one-sided label smoothing as a form of regularisation of the discriminative part of the process. This method avoids extremely confident classification which may lead to extremely extrapolating behaviour and the authors showed that one-sided label smoothing helps to reduce the propensity of the discriminator to be fooled by adversarial examples possibly prepared by the generator. This smoothing is done only on the real data samples in which the 0.9 probability is used instead of the correct probability of 1.

Minibatch Discrimination Researchers showed that training often leads to a poor diversity of the generated samples and the discriminator has only limited options to deal with this. The main idea of minibatch discrimination is to encourage the discriminator to focus also on the variance among samples.

Batch Normalisation Researchers showed insufficient convergence without some kind of batch normalisation [Salimans et al., 2016]. However, a simple batch normalisation can bring intra-batch correlations in the generated samples [Goodfellow, 2016].

At the same time, due to this, a new way to do the batch normalisation was introduced taking into account some disadvantages of the standard batch normalisation. The so-called *virtual batch normalisation* [Salimans et al., 2016] uses statistics collected on a reference batch of examples – these are chosen once and fixed at the beginning of training – and on just normalised batch itself.

Capacity Balancing There is clear intuition about the synergy between the generator and the discriminator. Let us suppose that the discriminator has a high capacity and each sample from the generator is classified as a fake one. In this particular scenario, the generator would be left without any gradient – the discriminator is saturated and the generator has no gradient as a source of information. It could happen if the discriminator can distinguish very well between real and fake before the generator can approximate the data distribution. However, we can deal with this using some tricks. The straightforward idea of lowering the discriminator capacity is one way. However, this may lead to worse performance at the end. Instead, [Goodfellow, 2016] suggests using the non-saturating game or one-sided label smoothing should be the better choice.

There are also known experiments with asymmetrical updates in the training of the discriminator and the generator. Empirically, it was shown that the discriminator supposed to have a higher capacity than the generator [Mescheder et al., 2018].

Gradient Penalty The recently proposed Wasserstein GAN (WGAN) makes progress towards stable training of GANs but sometimes can still generate only poor samples or fail to converge.

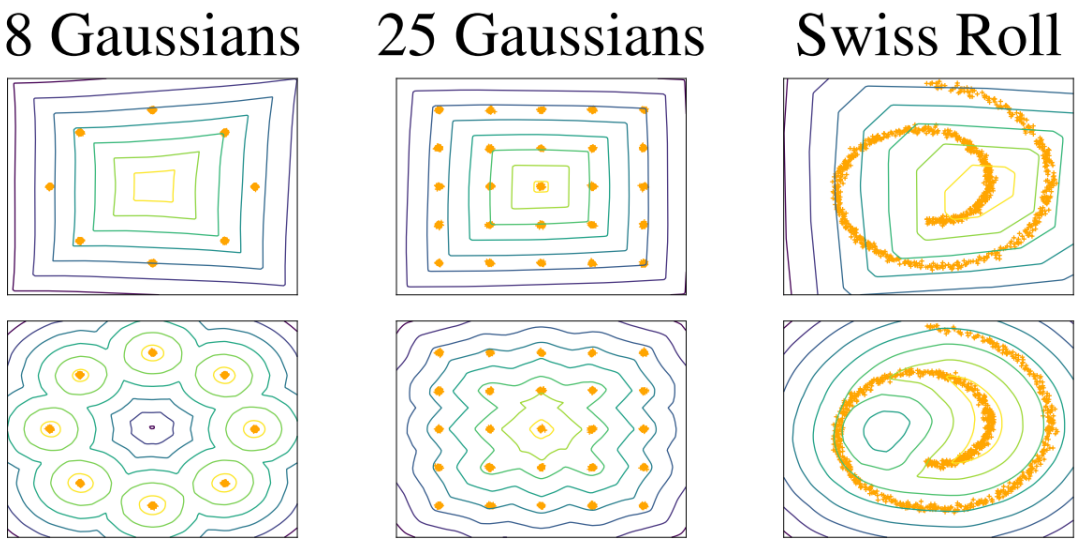
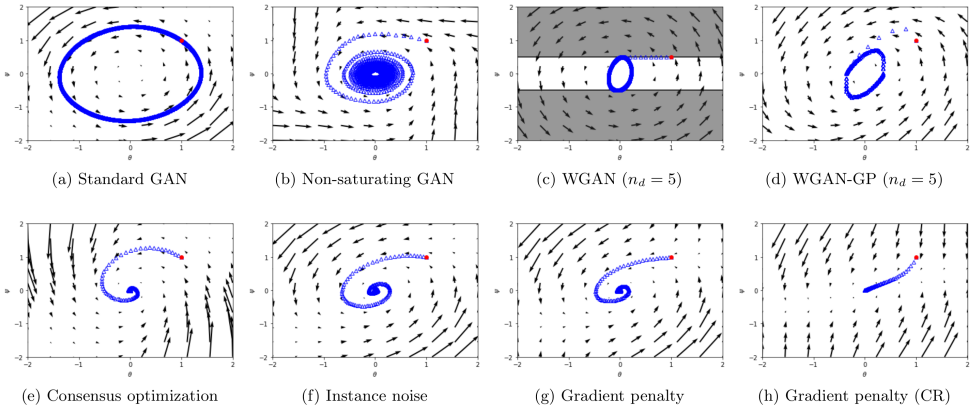


Figure 3.8: The WGAN and WGAN-GP trained to optimality on toy datasets. These setups demonstrate benefits of gradient penalty. top: standard WGAN; bottom: WGAN-GP. Image taken from [Gulrajani et al., 2017].

[Gulrajani et al., 2017] observed that under a weight-clipping constraint, the neural network architectures that try to attain their maximum gradient norm end up learning elementary functions. They also demonstrated this on toy distributions designed for this experiment and trained the WGAN critics with weight clipping to optimality, holding the generator distribution fixed on the real distribution with Gaussian noise added (Figure 3.8).

At the same time, they introduced a new method – gradient penalty. This approach adds a new term into standard WGAN objective and penalises unwanted gradient values. This provides better gradients with more information during the training and it is stable during the whole procedure (Figure 3.9). This approach was named as WGAN-GP.

$$\lambda \mathbb{E}_{x \sim \mathbb{P}_x} \left[(||\nabla_x D(x)||_2 - 1)^2 \right] \tag{3.40}$$



(a) Standard GAN (b) Non-saturating GAN (c) WGAN ($n_d = 5$) (d) WGAN-GP ($n_d = 5$)

(e) Consensus optimization (f) Instance noise (g) Gradient penalty (h) Gradient penalty (CR)

Figure 3.11: The image depicts different convergence processes of various GAN training algorithms. We see that whereas unregularised training of GANs and Wasserstein-GANs is not always convergent, training with instance noise or zero-centred gradient penalties leads to convergence. Image is taken from [Mescheder et al., 2018].

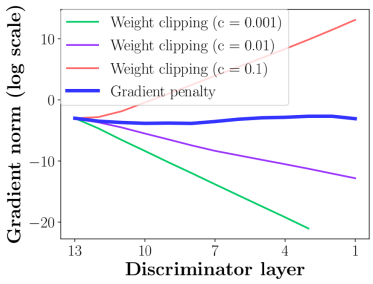


Figure 3.9: Gradient norms of deep WGAN critics during training on the Swiss Roll dataset either explode or vanish when using weight clipping but do not when using a gradient penalty [Gulrajani et al., 2017].

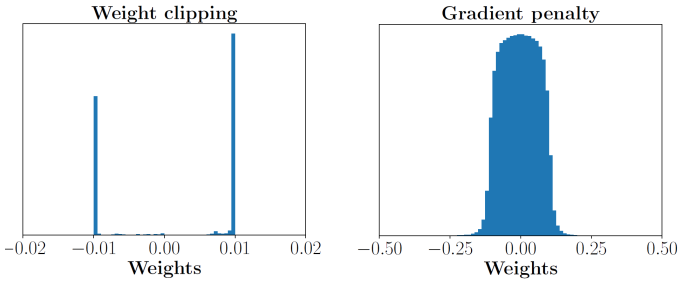


Figure 3.10: The image depicts the difference between distributions of weights in standard WGAN and WGAN-GP. The clipping pushes weights towards two values (the extremes of the clipping range). The gradient penalty does not suffer from this problem and provides better information to learn the model. Image was taken from [Gulrajani et al., 2017].

Instance noise The idea of adding noise into samples (both real and fake) reduces problem with orbit in two player game. This noise should bring a somewhat stochastic behaviour and corrupt the rotary motion [Sønderby et al., 2016, Mescheder et al., 2018].

3.4 GAN Architecture – Specific Extensions

Researchers experimented with a wide scale of modifications of generative adversarial architectures. The LAPGAN [Denton et al., 2015] uses hierarchical discrimination – it adds more interactions between G and D at different levels of abstraction.

Authors of InfoGAN add mutual information term into the mini-max game to disentangle the latent space [Chen et al., 2016]. They were able to train this unsupervised setup on different tasks in the image domain. VGAN [Wang et al., 2018a] adds variational methods into the standard

LSTM generator and via the GAN framework it trains the generator which dominates SeqGAN [Yu et al., 2017] on Amazon Food Reviews. TAC-GAN [Dash et al., 2017] shows a sentence-conditioned picture generation using the adversarial architecture.

The authors of Adversarial Variational Bayes [Mescheder et al., 2017] made deep analysis of combination of the VAE and the GAN architecture and suggest best practices for this setup.

Adversarial Autoencoder [Makhzani et al., 2015] is an interesting way which allows us to place discrete stochastic unit alongside with the discriminator; therefore, we can generate discrete samples like words without breking the gradient flow.

3.5 Importance of Generative Models in Various Tasks

The text generation task is related to various other tasks. There are lots of systems that need to generate text as their own output – image caption generation, machine translation, dialogue systems, question answering, etc.

In other cases, the generators doing a good job can affect the accuracy of other models in the same domain as artificial data for the data set extension. The problem is that creating the proper generator may be even a harder problem than creating a good classifier. The generator-discriminator scheme is the best example of the connection between these two tasks. If we had a generator with poor results to fool the discriminator, it is not challenging to distinguish between the real and the fake samples. In the other hand, we can create generators as good as the data set allows. In the case of the GAN architecture, we try to train the generator and the discriminator simultaneously which can be convenient for both tasks – generation and classification. The following paragraphs summarise the state-of-the-art in various tasks with highlighting the role of the generator.

3.5.1 Machine Translation

Machine translation is a process of transformation of a text from a source language into a target language. One possibility is to create a semantic representation of a sentence and generate the target sentence from this internal representation. This architecture is called *Encoder-Decoder* [Cho et al., 2014]. The main disadvantage is in a limited length of the translated text. The limited size of the internal representation can catch only limited semantics.

A significant benefit is brought by the attention mechanism which tries to use the original words in combination with the output from the encoder. The transformer model [Vaswani et al., 2017] shows that we actually do not need to build a representation of the whole sentence because the attention mechanism is sufficient.

[Wu et al., 2017] explored the potential of GAN in neural machine translation. [Yang et al., 2017] applied a CNN-based discriminator to the machine translation task with the novel BLEU-reinforced GAN.

3.5.2 Image Caption Generation

When we consider how the neural machine translation models work in general, the image label generation is a strongly related task. After we extracted semantics of the image into some kind of representation, we could generate a text in a natural language. From some point of view, we can see this as a transformation from an image representation of the scene into a description of the same scene in a natural language.

In order to create the scene representation and to capture the information, convolutional neural

network architectures are predominately used. After that, mechanisms very similar to the standard decoder known from the translation task are used [Vinyals et al., 2015]. [Xu et al., 2015] added the attention mechanism.

3.5.3 Text Summarization

The model by [Rush et al., 2015] was developed for sentence summarization. This is a simpler problem than full document summarization. The approach follows the general approach used in NMT – encoder and decoder. They tried out bag-of-words encoder, convolutional encoder, and attention-based encoder.

3.5.4 Language Modelling

Language modelling (LM) is one of the basic disciplines in natural language processing and it is a crucial component for generating a syntactically and semantically correct sequence of words. The ability to predict correct words conditioned by a sequence of previous ones is an integral part of various tasks. Modern deep models based on neural networks contain language modelling in their long stacks and often are not explicitly forced to perform it on a specific layer. However, we can have some basic intuition about where the LM is performed. In the standard encoder-decoder scheme (without attention) (see Sec. 3.5.1 on pg. 35) and regarding the fact that the complete information must be compressed and captured in the only one hidden state (assuming the correct output of our model), we can be pretty sure that language modelling must be done in the decoder.

In the very recent years, few papers on using the generator-discriminator scheme were published – MASKGAN [Fedus et al., 2018] and very similar Electra [Clark et al., 2020]. The goal is to use the generator to replace words instead of masking them[3]. In the Electra paper, different ways of training were used. Although the generator-discriminator scheme encourages to use of the GAN scheme, they were not able to overcome the maximum likelihood approach. The experiments showed a poor variability of the generated samples in comparison to maximum likelihood approach in the sense of accuracy of language modelling.

Few-Shot Learning with Language Models GPT-2 and GPT-3 came with a really unusual approach to various – usually supervised – tasks. Although it was not trained on different tasks, unexpectedly, it can solve some of them at elementary level using just conditioning a well-trained language model. This setup can solve a wide range of tasks like e.g. question answering (contextual and non-contextual), machine translation, common sense reasoning, text completion, even basic mathematical operations and more. The quality of the output fluctuates; however, it is far from random guessing. For more information see GPT-2 and GPT-3 on pg. 16 or the related papers [Radford et al., 2019, Brown et al., 2020].

3.5.5 Data Augmentation

Extending a small dataset by adding noise into existing examples or even generating new ones is desired ability in low resource tasks. A simple method for data augmentation is replacing random words in samples by randomly generated new ones from a dictionary with uniform probability [Wang et al., 2018b]. A more sophisticated way is to focus on those that have small TF-IDF assuming that such words are not crucial as features and noise can be added there [Xie et al., 2019]. [Wei and Zou, 2019] use synonym replacement, random swap, random deletion, or insertion. Generative models are promising when it comes to more sophisticated augmentation and a

[3]See the contrast with the BERT (see Sec. 2.5.2 on pg. 15)), RoBERTa [Liu et al., 2019], and other BERT-like approaches.

high-capacity generator can extend a training dataset in a much better way than the previously mentioned methods [Frid-Adar et al., 2018, Malandrakis et al., 2019].

3.5.6 Text Semantic Similarity

From the above described and discussed facts, it is evident that the generators play a significant role in many common tasks and improving them is worth the effort. Besides, as Electra showed[Clark et al., 2020], with the generator-discriminator scheme, we can overcome the state-of-the-art techniques on the language modelling task; despite this fact, there is still room for an improvement of adversarial training.

3.5.7 Image Domain Applications

Generative probabilistic models have the ability not only to create a new content; especially in the image domain, they also have a wide range of reconstruction-related applications including inpainting [Yeh et al., 2017, Yu et al., 2018], denoising [Yang et al., 2018], colourization [Nazeri et al., 2018], and super-resolution [Ledig et al., 2017].

3.5.8 Feature Visualization

Because of the complexity of the modern neural network models, it is not easy to understand how the models do what they do. There were several experiments that tried to force usual classifiers to generate some samples that should help our better understanding of how a neural network acts in such a model. The smart way – optimisation inputs w.r.t. specific neurons – enables us to see an image which would activate a specific neuron to maximum. [Olah et al., 2017] deal with different setups and show results of this applications of this method at different levels. They were successful in visualisation of the contribution of single neurons to the whole model behaviour.

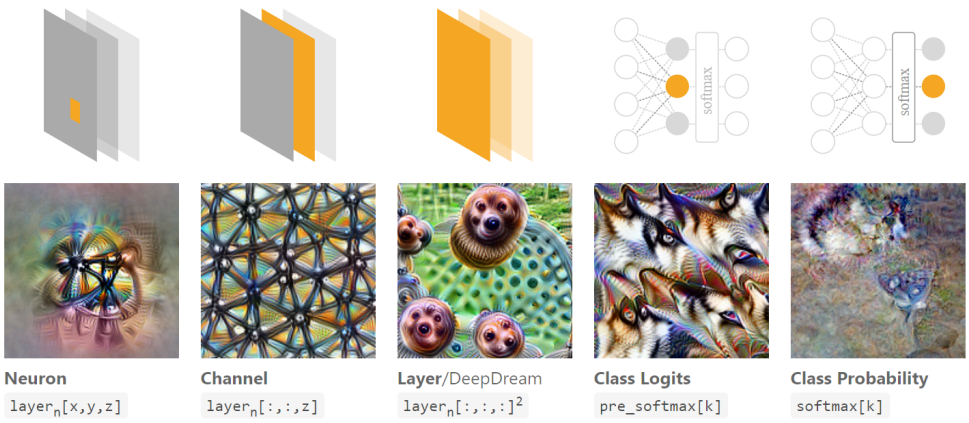


Figure 3.12: Feature visualisation by optimisation [Olah et al., 2017]

If we decide to optimise the input of the whole network with respect to the neurons indicating the resulting classes, we get an adversarial example which we have already mentioned before in sec. 3.3.11. This sample can be classified with high confidence as something completely different from what a human would decide to assign as a label.

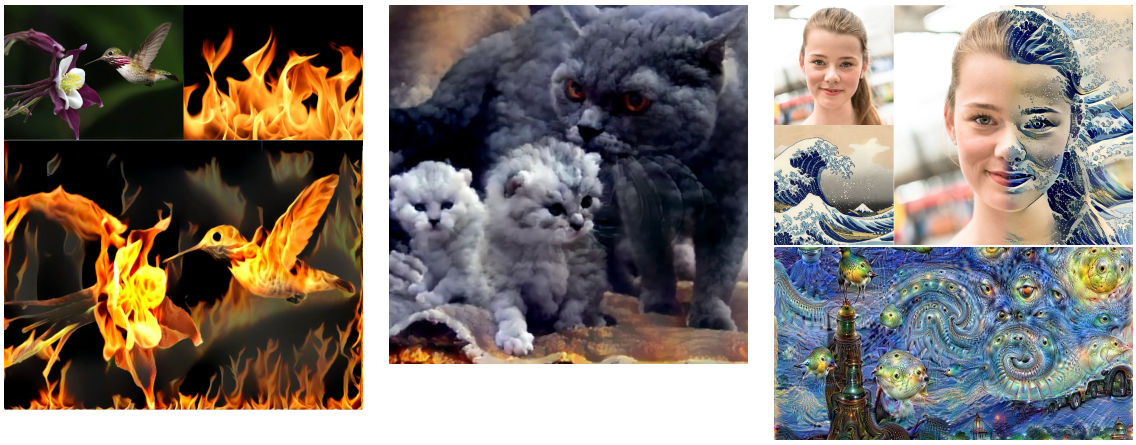


Figure 3.13: Examples presented at https://deepdreamgenerator.com/ [Olah et al., 2017]

Researchers are still exploring possible setups to get better understanding of models of neural networks [Wang et al., 2020, Carter et al., 2019].

Forcing high activation values in inner layers of deep networks yields also impressive outputs. On the top of this type of optimisation, Deep Minds project - The Deep Dream Generator was created[4]. It shows that in image domain, we can affect images in different ways – from style changes to something that can be seen as hallucinations. Also, open-source tools for doing similar experiments are publicly available[5].

3.6 Generation with Dependencies and Conditioning

So far, we have dealt with elementary mechanisms used in neural networks and more complex models built on these principles afterwards. We discussed just a little conditioning of those models to force them to generate text with some specific information. It is a more or less challenging task and it depends on the length and diversity of the wanted output, complexity and amount of the information we want to capture. In models with tractable density function, we can force the system to generate specific words by increasing probabilities to generate such words. With these changes, we definitely can force the system to generate some words more probably than the others.

We can also look at standard machine translation models and speak about conditioning. We have a text generator for a language we are translating into which receives some input information encoded by an encoder. The encoder reads a source sentence and encodes the information hidden in this text into some other representation of the information which the decoder can understand and can generate a text in another language. This information vector of numbers V_i should contain everything we want to translate into the other language. In fact, we can use this vector V_i (created for the task of translating $L_A -> (V_i) -> L_B$) for training of a system translating $L_A -> (V_i) -> L_C$ without recomputing the vector V_i. It is already a language-independent information representation. The problem is that it is not easy to connect vectors in this space with the real world without using an encoder. This vector space is somehow able to capture the information from the real world; however, finding human-understandable base vectors is not a straightforward mission.

Nonetheless, in some cases, we do not need to change the latent vector directly in a specific way. In

[4]https://deepdreamgenerator.com/
[5]https://github.com/openai/lucid

the case of image caption generation, it does not make any sense to generate captions from messed up inner representations without any connection to the original images. Alternatively, machine translation mentioned before is a similar example.

There, nevertheless, may be use cases when we need just *some* artificial sample. In image domain, it is well-known in the task of random face generation which can be handy for e.g. for generating random and non-existing faces in some pictures or videos taken on streets to preserve peoples' privacy; in the text domain, the generation of artificial textual data for training generative adversarial models and increasing the accuracy of classifiers on real-data tasks.

Chapter 4

Previous Work

The following chapter summarises my work during the course of previous three years. A link to my aims in a doctoral thesis is added at the end of each part as [AIM 1,2,3]

English Dataset For Automatic Forum Extraction At the beginning of my Ph.D. study, I focused on the processing of tree structures using deep learning methods.

We described the process of collecting, maintaining, and exploiting an English dataset of web discussions[Sido et al., 2019a]. The dataset consists of many web discussions with hand-annotated posts in the context of the web page tree structure. Each post consists of a username, date, text, and quotations used by its author. The dataset contains 79 different websites with at least 500 pages from each. Each web page include a tree structure of HTML tags with texts taken from selected web pages. We also describe algorithms trained on the dataset, in the paper. The algorithms employ basic architectures (such as a bag of words with SVM classifier and LSTM network) to set a dataset baseline.

We were also working on modifications of the standard LSTM unit for tree structure processing. The task was intended as information retrieval from tree structures. However, with the rise of approaches dealing with flattened tree structures and their performance, we have decided not to follow this way anymore. The work in this field helped me to understand text processing using neural networks and gave me a better notion of dataset building. [AIM1]

On Injecting Entropy-like Features Into Deep Neural Networks for Content Relevance Assessment Before the Transformer-like and Bert-like models were popularised, we pursued an approach based upon adding large-scale quality measure into deep neural models. We injected a global (or generally large-scale) quality measure into a deep neural network (DNN) in order to compensate for the tendency of DNNs to found the resulting classification virtually from a superposition of local neighborhood transformations and projections. We used a *state probability-like feature* as the global quality measure and injected it into a DNN-based classifier deployed in a specific task of determining which parts of a web page are of certain interest for further processing by NLP techniques. Our goal was to decompose web sites of various internet discussion fora to useful content, i.e. the posts of users, and useless content, i.e. forum graphics, menus, banners, advertisements, etc. This work has not been published yet.

I stopped my work on this topic, because my supervisor transferred me to a newly approved grant (no. TL02000288). Nevertheless, this work enabled me to get acquainted myself with deep learning methods used for processing textual data. Since all my aims of the doctoral thesis lay in the text domain, this work correlates with all of them but mostly with the first two. [AIM1,2]

Curriculum Learning We also investigate the phenomenon called *curriculum learning* for deep neural network models. Our research is focused on models for the sentiment analysis task. We designed a new curriculum learning method for textual data. It reorders the training dataset to first introduce more straightforward examples into the training process. We estimate the difficulty of the samples by measuring lengths of the sentences. Simple examples are supposed to be shorter. We also experimented with measuring frequencies of words, which is a technique designed by previous researchers. We attempted to evaluate changes in the overall accuracy of the models using both curriculum learning techniques. Our experiments did not show an increase of accuracy for any of the methods. Nevertheless, we reached a new state of the art in sentiment analysis for the Czech language as a by-product of our effort.

We achieved the best results with the RNN model. It is able to reach $80.5\% \pm 0.155$ on 95% confidence interval what is a new state of the art on this data set. The CNN model achieved $78.7\% \pm 0.245$. These results were obtained on the test part of the original dataset by running a fixed number of epochs and averaging scores over 10 runs. This work was published in the research paper [Sido and Konopík, 2019].

Even though we could not profit from the curriculum strategy with a neural network model, we were able to outperform the previous approaches. However, curriculum learning can help to make GANs more stable – splitting training process into more epochs with different levels of richness of language. And likewise, employ generative models for data augmentation is more sophisticated than simple reordering data samples in general curriculum learning schema. Consequently, it could possibly lead to more robust and more accurate models. [AIM1,2].

Deep Learning for Textual Data on Mobile Devices As with many other powerful tools, AI brings many advantages but many risks as well. Predictions and automation can significantly help in our everyday lives. However, sending our data to servers for processing can severely affect our privacy. We described experiments designed to find out whether we can enjoy the benefits of AI in the privacy of our offline mobile devices. We focused on textual data since these are easily stored in large quantities for mining by third parties. We measured the performance of deep learning methods in terms of accuracy (compared to fully-fledged server models) and speed (number of text documents processed in a second). We concluded our paper with findings that with few relatively small modifications, mobile devices can process hundreds to thousands of documents. This work was published in the research paper [**?**].

This work uses a neural network classifier, since one of my aims in the doctoral thesis is improving the performance of conventional models. It is work which brought me a better basic knowledge of the respective area.[AIM1]

SEMEVAL 2020 In 2020, we participated in the SEMEVAL 2020 competition working on the semantic-change task. We examined semantic differences between specific words in two corpora, chosen from different time periods, for English, German, Latin, and Swedish. Our method was created for the SemEval 2020 Task 1: Unsupervised Lexical Semantic Change Detection. We were ranked 1^{st} in Sub-task 1: binary change detection, and 4^{th} in Sub-task 2: ranked change detection. Our method is fully unsupervised and language-independent. It consists of preparing a semantic vector space for each corpus, earlier and later; computing a linear transformation between earlier and later spaces, using Canonical Correlation Analysis and Orthogonal Transformation; and measuring the cosines between the transformed vector for the target word from the earlier corpus and the vector for the target word in the later corpus.

Our participation in the SEMEVAL 2020 competition was a highly collaborative work. My responsibility was to explore the possibility of using the LDA model for this task. Furthermore, I was also participating in creating a development dataset. [AIM1]

News Generation The use of automated journalism became an established practice in English-speaking countries less than ten years ago. Narrative Science and Automated Insights developed creative software that automatically generates reports. Several media outlets, including The Associated Press (AP), have started to publish the reports generated this way. Media landscape barriers based upon Slavic languages, such as Czech, have caused some delays in introducing automated journalism.

We created a case study of the application of algorithms that transform large data files into news texts in The Czech News Agency (ČTK). Our team prepared algorithms that generate reports on trading results at the Prague Stock Exchange without human intervention for The Czech News Agency in 2019.

The study deals with the production of algorithms and compares the speed of generation of messages generated by humans against the algorithms and examines their quality. Our research also used observations and questionnaire surveys of selected journalists and editors who work with reports from the Prague Stock Exchange. The article also provides the opinions of journalists of The Czech News Agency on the application of automated journalism and artificial intelligence journalism in their newsrooms.

This software runs on our servers to provide the automatic news service. The only purpose is to bring an automated, unbiased, fast, and in-depth analysis of the Prague Stock Exchange in a textual form ready to be published right away. However, rigid rules and demands on the resulting text forced us to use a rule-based approach. For more details, see the published paper [Moravec et al., 2020]. AIM2,3

During the same project, we are supposed to generate summaries of Czech news. We have already tried several approaches to this task. However, news–summary pairs from the Czech News Agency showed up to be rather complicated. They contain general knowledge and strong dependency in time. An end-to-end approach turned out to be impossible for this purpose. We decided to use a way of clustering sentences to group similar pieces of information. The next step will be presenting these clusters in their short forms while preserving time dependencies as consistent news. Journalists are supposed to interact with this representation to produce a final report ready for publication. AIM1,2,3

Czech Dataset for Semantic Textual Similarity For the purpose of grouping semantic clusters, we need a model for creating the representations of sentences. Besides, the domain of news is narrow and there are not many datasets that could support this task. As a consequence, we decided to create a new Czech dataset for semantic textual similarity. For this, we use our collaboration with the Charles University and its students of journalism. We prepared a web application which enables students to annotate 123 813 pairs of sentences. This study is in the second phase of validation by another group of students. As of now, we are preparing a paper about the collection of this dataset, and about the basic experiments. We suppose the dataset to be published in 2021. AIM1,2,3

Czech Model for Semantic Representation We also noticed that English and multilingual models, in general, are limited in tasks in one specific language and can be outperformed by monolingual ones. For this reason, we decided to train the Czech model based upon the BERT architecture. The paper describes a process of training of a Czech semantic model based upon a BERT-like approach. In the first part, the data preparation and the pretraining of such a model are mentioned. Due to high computational requirements, the process is computed on the Czech GPU cluster. The second part is dedicated to the evaluation of this model on various tasks. The results of our work will be publicly accessible on GitHub and in the Hugging face repository for non-commercial experiments and applications.

We showed that our Czech model outperformed other multilingual models in the majority of our

experiments. This work is ready to be published.

The work on the described Czech model was a highly collaborative effort. My responsibility was to prepare the dataset, handle the pretraining process, and evaluate this model on the downstream task of semantic textual similarity. [AIM1,2,3]

Conclusion and Aims of Doctoral Thesis

An extensive research of generative models was done on continuous space and we have seen various applications on image tasks. Text generation task is more challenging because of the problem with the differentiation of the sampling step for outputting discrete tokens. Also, even a small error in the generated sample is glaring for humans.

In the last years, generative models came in useful in many domains. It was shown that they can improve the performance of standard models, and we can look at them as at zero-shot learning applications in some cases (GPT-2,GPT-3).

In this work, the basic principles of the state-of-the-art neural networks were introduced. Upon these fundamental principles, more complex models were established. In the scope of the project TL02000288, these models will be used for generating news.

Note 4: Transformation of Journalism Ethics in the Advent of Artificial Intelligence

Transformation of Journalism Ethics in the Advent of Artificial Intelligence – TL02000288 – is a project funded by TAČR –Technology Agency of the Czech Republic. The goal of this project is to apply methods of artificial intelligence in the journalistic domain. Our role in this project is to help with analysing standard journalists' work and provide some automatic assistance in everyday labour. Simple examples of such an assistance are an automatic generator of short news from narrow domains e.g. the stock exchange and analysis of a large amount of news and helping with their summarisation.

In the text generation domain, scientific research has shown remarkable results in the last years, though, we are sitll far from sufficiently useful samples. We also introduced models that could potentially profit from high-capacity generators.

The majority of models is developed for less flectional languages and is thus unsatisfactorily evaluated on more complex natural languages. Previous research has shown that there is an issue coming up from the combination of more complex conditioning and low resources in not so widespread languages.

Application of Generative Models in Journalistic Domain At the beginning of our collaboration with Czech journalists, I have already met some limitations during our effort to apply NLP methods in real operation conditions. Right now, we host first real application which arose from the project TL02000288 and the only purpose is to bring automatic, unbiased, fast and deep analysis of the Prague Stock Exchange in textual form ready to be published right away. This application is being actively exploited by several news agencies and the Prague stock exchange publishes reports about stock trading every day on their website[1]. A basic study of the whole

[1] www.e15.cz;www.ctk.cz;www.pse.cz

process of defining, implementing and deploying this automatic system for generating news from the Prague stock exchange was published [Moravec et al., 2020]. After the cooperation on project TL02000288, I have already acquired better understanding of journalists' requirements. A use of more complex and advanced models as a replacement of humans in journalistic work is limited to narrow domains and remains still challenging due to high demand on strict fact reasoning. However, there is a space for automatising of reasonably simple processes in the news production from domains like the stock exchange, weather forecast, traffic, etc.

Semi-Supervised Learning Modern algorithms require many labelled examples to be able to generalise them. If we consider a real-world scenario – we have many data samples from a given domain but labels for many or even most of the training examples are missing – data augmentation and semi-supervised learning are one of the possible strategies, and they are a hot topic.

Aims of Doctoral Thesis Based on my previous research, the doctoral thesis will be focused on the following goals:

1. Adapt the current generative or generator-discriminator methods and try to improve the performance of conventional models used for building representations of semantics in the textual domain.
2. Design a new method for adding constraints into the generative model architecture for the text domain.
3. Experiment with generative or generative-adversarial models in the journalistic domain on the Czech language with a focus on including specific entities, facts, etc.

Acknowledgement

This work has been partly supported by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

Bibliography

[att, 2016] (2016). Attention and augmented recurrent neural networks. `https://distill.pub/2016/augmented-rnns/`. Accessed: 2019-07-01.

[fea, 2017] (2017). Feature visualization. `https://distill.pub/2017/feature-visualization/`. Accessed: 2019-07-01.

[att, 2019] (2019). Attn: Illustrated attention. `https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3`.

[Achiam, 2018] Achiam, J. (2018). Spinning Up in Deep Reinforcement Learning.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Baraldi et al., 2015] Baraldi, L., Grana, C., and Cucchiara, R. (2015). A deep siamese network for scene detection in broadcast videos. *arXiv preprint arXiv:1510.08893*.

[Baziotis et al., 2017] Baziotis, C., Pelekis, N., and Doulkeridis, C. (2017). Datastories at semeval-2017 task 6: Siamese lstm with attention for humorous text comparison. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 390–395.

[Bellemare et al., 2016] Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479.

[Bellemare et al., 2017] Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org.

[Bengio et al., 1994] Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Bouchacourt et al., 2018] Bouchacourt, D., Tomioka, R., and Nowozin, S. (2018). Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Bromley et al., 1993] Bromley, J., Bentz, J., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Sackinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. *Int.J. Pattern Recognit. Artif Intell*, 7.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Carter et al., 2019] Carter, S., Armstrong, Z., Schubert, L., Johnson, I., and Olah, C. (2019). Activation atlas. *Distill*. https://distill.pub/2019/activation-atlas.

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[Chen et al., 2016] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

[Chiu and Nichols, 2016] Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Clark et al., 2020] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

[Coquelin and Munos, 2007] Coquelin, P.-A. and Munos, R. (2007). Bandit algorithms for tree search. *arXiv preprint cs/0703062*.

[Dai et al., 2015] Dai, J., He, K., and Sun, J. (2015). Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643.

[Dash et al., 2017] Dash, A., Gamboa, J. C. B., Ahmed, S., Liwicki, M., and Afzal, M. Z. (2017). Tac-gan-text conditioned auxiliary classifier generative adversarial network. *arXiv preprint arXiv:1703.06412*.

[Dauphin et al., 2017] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

[Deco and Brauer, 1995] Deco, G. and Brauer, W. (1995). Higher order statistical decorrelation without information loss. In *Advances in Neural Information Processing Systems*, pages 247–254.

[Denton et al., 2015] Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dinh et al., 2016] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

[Doersch, 2016] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[Donahue and Rumshisky, 2018] Donahue, D. and Rumshisky, A. (2018). Adversarial text generation without reinforcement learning. *arXiv preprint arXiv:1810.06640*.

[Faruqui and Dyer, 2014] Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

[Fedus et al., 2018] Fedus, W., Goodfellow, I., and Dai, A. M. (2018). Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

[Frey et al., 1996] Frey, B. J., Hinton, G. E., and Dayan, P. (1996). Does the wake-sleep algorithm produce good density estimators? In *Advances in neural information processing systems*, pages 661–667.

[Frid-Adar et al., 2018] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE.

[Gehring et al., 2017] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

[Gelly et al., 2006] Gelly, S., Wang, Y., Munos, R., and Teytaud, O. (2006). *Modification of UCT with patterns in Monte-Carlo Go*. PhD thesis, INRIA.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Goodfellow, 2016] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Graves et al., 2013] Graves, A., Jaitly, N., and Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE.

[Graves et al., 2014] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hsieh et al., 2018] Hsieh, J.-T., Liu, B., Huang, D.-A., Fei-Fei, L. F., and Niebles, J. C. (2018). Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526.

[Huang et al., 2015] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Hyvärinen and Pajunen, 1999] Hyvärinen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439.

[Iyyer et al., 2015] Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Jin et al., 2017] Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., and Fang, Z. (2017). Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*.

[Kim et al., 2016] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

[Kusner and Hernández-Lobato, 2016] Kusner, M. J. and Hernández-Lobato, J. M. (2016). Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

[Lang et al., 1990] Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[Li et al., 2019] Li, Y., Pan, Q., Wang, S., Peng, H., Yang, T., and Cambria, E. (2019). Disentangled variational auto-encoder for semi-supervised learning. *Information Sciences*, 482:73–85.

[Ling et al., 2015] Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

[Liu et al., 2016] Liu, Y., Chen, X., Liu, C., and Song, D. (2016). Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Luong et al., 2013] Luong, T., Socher, R., and Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

[Makhzani et al., 2015] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

[Malandrakis et al., 2019] Malandrakis, N., Shen, M., Goyal, A., Gao, S., Sethi, A., and Metallinou, A. (2019). Controlled text generation for data augmentation in intelligent artificial agents. *arXiv preprint arXiv:1910.03487*.

[Mescheder et al., 2018] Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Miller, 1998] Miller, G. (1998). *WordNet: An electronic lexical database*. MIT press.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

[Mnih et al., 2014] Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Moravec et al., 2020] Moravec, V., MackovÁ, V., Sido, J., and Ekštein, K. (2020). The robotic reporter in the czech news agency: Automated journalism and augmentation in the newsroom. *Communication Today*, 11(1).

[Mueller and Thyagarajan, 2016] Mueller, J. and Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Nazeri et al., 2018] Nazeri, K., Ng, E., and Ebrahimi, M. (2018). Image colorization using generative adversarial networks. In *International conference on articulated motion and deformable objects*, pages 85–94. Springer.

[Neculoiu et al., 2016] Neculoiu, P., Versteegh, M., and Rotaru, M. (2016). Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157.

[Olah et al., 2017] Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*. https://distill.pub/2017/feature-visualization.

[Oord et al., 2016a] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[Oord et al., 2016b] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

[Oord et al., 2017] Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., et al. (2017). Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*.

[Paine et al., 2016] Paine, T. L., Khorrami, P., Chang, S., Zhang, Y., Ramachandran, P., Hasegawa-Johnson, M. A., and Huang, T. S. (2016). Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Prakash et al., 2016] Prakash, A., Hasan, S. A., Lee, K., Datla, V., Qadir, A., Liu, J., and Farri, O. (2016). Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

[Reed et al., 2016] Reed, S., van den Oord, A., Kalchbrenner, N., Bapst, V., Botvinick, M., and De Freitas, N. (2016). Generating interpretable images with controllable structure.

[Rush et al., 2015] Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

[Salakhutdinov and Hinton, 2009] Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Sido and Konopík, 2019] Sido, J. and Konopík, M. (2019). Curriculum learning in sentiment analysis. In *International Conference on Speech and Computer*, pages 444–450. Springer.

[Sido et al., 2019a] Sido, J., Konopík, M., and Pražák, O. (2019a). English dataset for automatic forum extraction. *Computación y Sistemas*, 23(3).

[Sido et al., 2019b] Sido, J., Konopík, M., and Reismüllerová, J. (2019b). Deep learning for text data on mobile devices. In *Applied Electronics*, pages 147–155.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

[Sønderby et al., 2016] Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2016). Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*.

[Srivastava and Salakhutdinov, 2012] Srivastava, N. and Salakhutdinov, R. R. (2012). Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230.

[Tai et al., 2017] Tai, Y., Yang, J., and Liu, X. (2017). Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155.

[Tang et al., 2014] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

[Tesauro, 1994] Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219.

[Tsitsiklis and Van Roy, 1997] Tsitsiklis, J. N. and Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081.

[Varior et al., 2016] Varior, R. R., Haloi, M., and Wang, G. (2016). Gated siamese convolutional neural network architecture for human re-identification. In *European conference on computer vision*, pages 791–808. Springer.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

[Wang et al., 2018a] Wang, H., Qin, Z., and Wan, T. (2018a). Text generation based on generative adversarial nets with latent variables. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 92–103. Springer.

[Wang et al., 2018b] Wang, X., Pham, H., Dai, Z., and Neubig, G. (2018b). SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

[Wang et al., 2020] Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., and Chau, D. H. (2020). Cnn explainer: Learning convolutional neural networks with interactive visualization. *arXiv preprint arXiv:2004.15004*.

[Watkins, 1989] Watkins, C. J. C. H. (1989). Learning from delayed rewards.

[Wei and Zou, 2019] Wei, J. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

[Wu et al., 2017] Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2017). Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

[Xie et al., 2019] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation for consistency training.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

[Yang et al., 2018] Yang, Q., Yan, P., Zhang, Y., Yu, H., Shi, Y., Mou, X., Kalra, M. K., Zhang, Y., Sun, L., and Wang, G. (2018). Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE transactions on medical imaging*, 37(6):1348–1357.

[Yang et al., 2017] Yang, Z., Chen, W., Wang, F., and Xu, B. (2017). Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

[Yeh et al., 2017] Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N. (2017). Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493.

[Yin et al., 2016] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

[Yu et al., 2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514.

[Yu et al., 2017] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Zhou et al., 2016] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

[Zhou et al., 2010] Zhou, S., Chen, Q., and Wang, X. (2010). Discriminative deep belief networks for image classification. In *2010 IEEE international conference on image processing*, pages 1561–1564. IEEE.