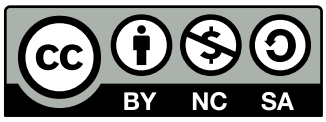
Aalto University
School of Science

Bachelor's Programme in Teknistieteellinen kandidaattiohjelma

Exploring Large Language Models for Clinical Code Assignment

Lasse Kukkula

Bachelor's Thesis
2025



Aalto University
School of Science

Author Lasse Kukkula

Title Exploring Large Language Models for Clinical Code Assignment

Degree programme Bachelor's Programme in Science and Technology

Major Computer Science

Teacher in charge Prof. Lauri Savioja

Advisor Prof. Alex Jung

Date 11.9.2025 **Number of pages** 24 **Language** English

Abstract

Clinical coding, the act of translating clinical transcripts into standardised diagnostic and procedural codes, is an essential statistics and billing tool in hospitals around the world. Despite it carrying a lot of weight in the capitalist healthcare system for billing purposes and for statistical and quality assurance purposes in socialised healthcare systems. The process of clinical coding is labour intensive and error prone.

Recent advances in large language models pave a way toward fully or partially automated reliable clinical coding. The research on large language models is constantly evolving and developing into new avenues that could and should be explored in the context of clinical coding.

This thesis provides a survey of the advancments in large language model architecture from the almost ancient multilayer perceptron and convolutional neural networks all the way to the current cutting edge advancements such as transformers and state space models. These findings are then taken into the context of clinical coding and some suggestions on future directions in the research into the use of large language models in clinical coding will be presented.

Keywords machine learning , large language models , artificial intelligence , clinical coding , ICD

Aalto University
School of Science

Tekijä Lasse Kukkula

Työn nimi Laajojen kielimallien tutkiminen tautiluokittelussa

Koulutusohjelma Teknistieteellinen kandidaattiohjelma

Pääaine Tietotekniikka

Vastuuopettaja Prof. Lauri Savioja

Työn ohjaaja Prof. Alex Jung

Päivämäärä 11.9.2025 **Sivumäärä** 24 **Kieli** englanti

Tiivistelmä

Tautiluokittelu, eli potilasasiakirjojen kääntäminen standardoituihin diagnoosi- ja toimenpidekoodeihin, on olennainen tilastointi- ja laskutusväline sairaaloissa kautta maailman. Kapitalistisissa terveydenhuoltojärjestelmissä se painottuu erityisesti laskutukseen, kun taas sosiaalisesti rahoitetuissa järjestelmissä se palvelee tilastointia ja laadunvarmistusta. Prosessi on kuitenkin työläs ja altis virheille.

Viimeaikaiset edistysaskeleet suurissa kielimalleissa (LLM) avaavat tietä luotettavalle tautiluokittelun täys- tai osittaiselle automatisoinnille. LLM-tutkimus kehittyy jatkuvasti ja tarjoaa uusia lähestymistapoja, joita kannattaa hyödyntää tautiluokittelussa.

Tämä kandidaatintyö käy läpi kielimallien arkkitehtuurin kehityskulun aina lähes “muinaisista” monikerroksisista perceptroneista ja konvoluutioneuroverkoista nykypäivän huipputeknologioihin, kuten transformereihin ja state-space malleihin. Tulokset asetetaan tautiluokittelun viitekehykseen, ja työn lopuksi esitetään suuntaviivoja tulevalle tutkimukselle suurten kielimallien hyödyntämiseksi kliinisessä koodauksessa.

Avainsanat koneoppiminen laajat kielimallit tekoäly tautiluokittelu ICD

Contents

1 Introduction

> "Imagine a world where a simple, yet precise, algorithm could transform chaotic medical records into organised, actionable insights in seconds. This is the promise of large language models (LLMs), cutting-edge artificial intelligence systems, poised to revolutionise the field of clinical code assignment. By exploring the integration of LLMs in healthcare, we stand on the brink of a breakthrough that could enhance patient care, streamline medical processes, and ultimately save lives." [1]

The amount of research into AI has risen nearly 2.5-fold between 2010 and 2022 [2] with the increased use of large language models (LLM), and in general is in general needed artificial intelligence (AI) applications. This increase in research leads to a rise in the number of LLM applications which in turn increases the use of AI creating a self-feeding cycle. Through this cycle LLMs are becoming better in text generation, image ' classification, and text classification. Many tasks, which were previously thought to be impossible for AI to achieve, are becoming increasingly trivial for LLMs to solve. As an example, GPT-4 has already outperformed humans on in some sections of the United States Medical Licensing Examination [3]. In addition, the very beginning of this introduction, the opening quote, was fully generated by GPT-4o [1]. These examples show the capabilities of LLMs in this day and age.

The research into AI in the medical field is centred on the diagnostic side of medicine [4]. For example, aid in the interpretation of magnetic resonance imaging, or computer tomography results. Following the research, a convolutional neural network model by Tian et. al. [5] achieved a 7% increase in diagnostic accuracy compared to medical doctors when diagnosing from X-ray images. These results are starting to show the capabilities of different AI applications in the diagnostic domain. The capabilities are raising a new question, "Where else in the medical field could ML methods be exploited?" One example where it could be used is the task of clinical coding.

Clinical coding is used to classify transcripts of a hospital visit into diagnostic and procedure codes [6]. This classification process aids in statistical analysis and billing within the hospital. The task of clinical coding takes a lot of work as one must read patient's file and then come up with codes matching it from thousands of possible codes [7]. The automation of this task would save a lot of man-hours. This task can be simplified to machine learning (ML) terms as a natural language processing (NLP) classification task. Some articles [8–10] have trained machine learning models with various results.

This thesis explores the use of large language models in the assignment of clinical codes. The theoretical background is first be presented with some literature review on the research into use of LLMs in automating clinical coding. The future of the automation of clinical coding with LLMs is finally discussed.

2 Automated Clinical Coding

2.1 Large Language Models

Natural language processing (NLP) is an important topic of research in machine learning, as a vast majority of communications are made through different languages. For this purpose, large language models (LLM) have been developed to model languages numerically. The quantification of language using such models is used in variety of applications such as machine translation, text generation, and text classification [11]. LLMs are becoming a part of everyday life with the developments in generative models such as ChatGPT, GitHub-CoPilot, and others. The generative power of these models is being utilised in various fields [12]. When talking about an LLM it usually employs a certain architecture, which combines various types of neural networks to create a complete model. In this section of this thesis, I will introduce some of these building blocks that enable LLMs to perform tasks.

2.1.1 Multilayer perceptron

A multilayer perceptron (MLP) [13] or more often vanilla neural network (NN) is the simplest type of feedforward neural network built from NN layers. It consists most often of an input layer, one or multiple hidden layers, and an output layer. Where each layer is usually a mathematical calculation of the form

$$f(x) = g(Wx + B) \tag{1}$$

where W is a weight matrix B is the bias matrix, x is the input matrix and g is a function often a sigmoidal function or a linear function such as ReLu [14]. During training W and B are being optimized to produce the best estimated output wiht respect to the actual outputs. Figure 1 [15] shows how information flows through an MLP.

MLPs are often good for some classification tasks [16] and are among the simplest of NN architectures. When it comes to natural language processing, the scalability of an MLP is poor due to its short context window as for a 1'500-word input with each individual word encoded as a vector belonging to $\mathbb{R}^{1000}$ the input matrix alone would be 1'500'000 elements. When weights would be considered, the number of parameters increases through each added layer resulting in the increase of the number of floating point operations per training iteration. Thus making the MLP a computationally expensive model for NLP.

In addition to computational expense, a major limitation of MLPs in NLP is the size of its context window. The context window is the number of tokens an LLM can process at a given time step. Due to the way in which MLP works, it would have to have the entire text fed into it at a time. Because of the nature of natural language different texts have different lengths and thus the input text would have to be either padded or truncated during preprocessing to fit the model. The model would have to be trained on an entire corpus of padded text. For example, for a MLP with a context window of 1'000'000 words reading the King James Bible, it being 738'000 words [17], would

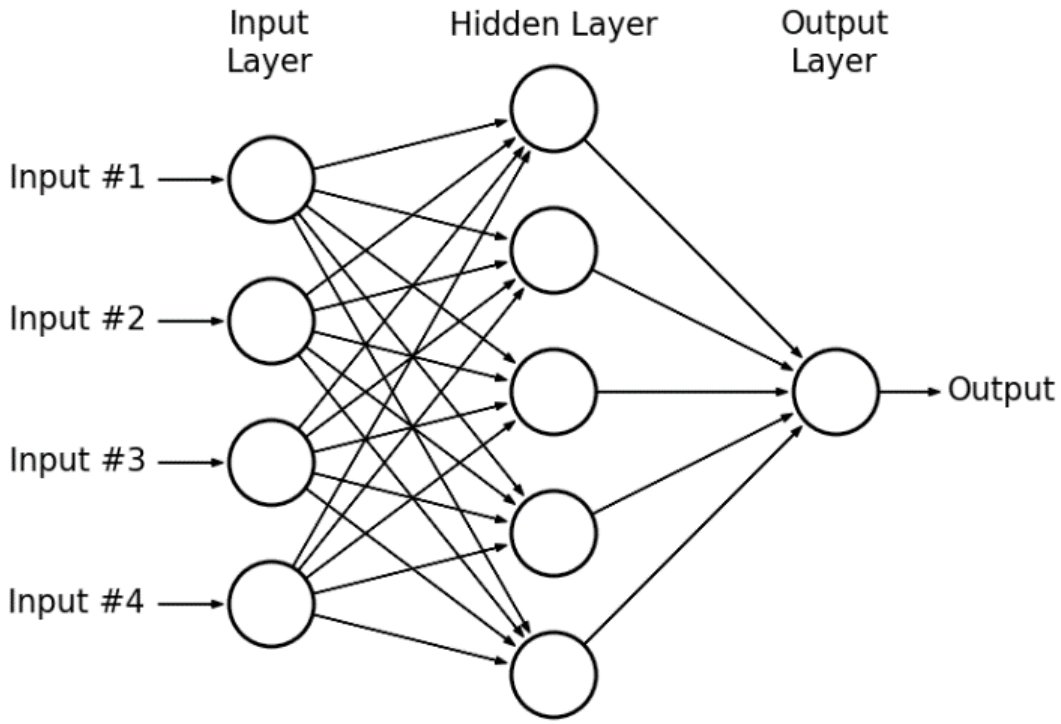


Figure 1: Multilayer Perceptron

require 262'000 words of padding resulting in a total of 262'000'000 extra floating point elements using 1GB of extra memory just to store the padded text.

2.1.2 Convolutional Neural Network

Convolutional neural network is a type of feed-forward network which utilises the convolution operator. CNNs reduce the computational expense when comparing to a MLP by utilizing the convolution operator, which for arrays $I \in \mathbb{R}^{M \times N}$ and $W \in \mathbb{R}^{m \times n}$ where $M \geq m$ and $N \geq n$ is defined as

$$C(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} W(k,l) I(i+k-1, j+l-1) \tag{2}$$

where $C \in \mathbb{R}^{(M-m+1)\times(N-n+1)}$ where C is the result of the convolution [18]. Figure 2 [19] shows an example of a 1D convolution.

Convolution allows the size of the weight matrix be smaller and thus reducing the number of floating point operations per training iteration. It also allows for the mixing of nearby inputs, which defines a context window smaller than the entire input. This allows for the model to be more scalable.

Like with MLPs CNNs require the inputs to be of the same size requiring either padding, causing increased computational load, or truncation, causing loss of information. This is a major limitation of pure CNN implementations in NLP.

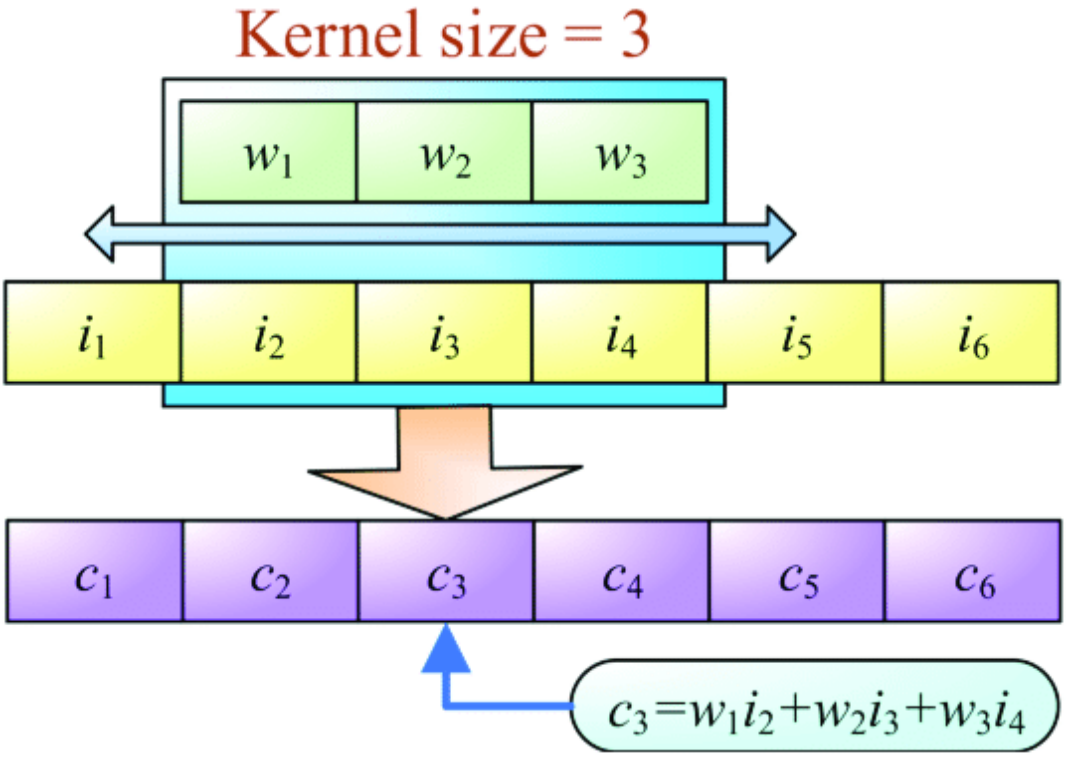


Figure 2: Example of a 1d convolution operation

2.1.3 Recurrent Neural Network

Unlike in a traditional feed-forward network [13] where the forward passes for two separate inputs are independent, recurrent neural network (RNN) [20] uses a hidden state allowing a state to be affected by each input before it. Hence, they are most often used for different kinds of time-series data, such as text processing and generation, audio processing, and video processing.

RNN works on a simple principle which is the recurrence unit (figure 3). A single layer recurrence unit can be described with the equation,

$$h_t = \sigma(W_{hh}x_t + W_{hh}h_{t-1} + B_h) \tag{3}$$

where Ws are the weights, x is the input matrix, h is the hidden state, σ is a non-linear function often sigmoid or hyperbolic tangent, and t is the timestep. As h_t is a function of h_{t-1}, we know that the nature of the formula is recurrent, and we know that the hidden state at time-step t is dependent on all inputs before it.

The use of a RNN improved the performance of text classification compared to MLP or CNN [21]. However, the RNN architecture has a significant drawback, for during the backpropagation the model often experiences vanishing or exploding gradients. This means that during gradient descent the weights either get stuck at local minima or start to diverge to infinity [22].

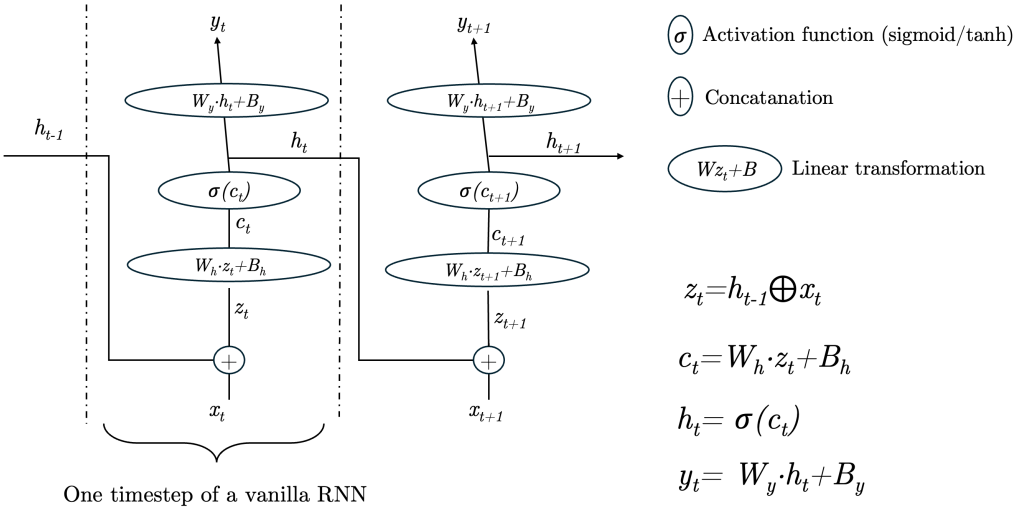


Figure 3: Recurrent Neural Network cell and equations

2.1.4 Residual Neural Network

Residual Neural Network is a type of feed-forward network which includes skip-connections to reduce the vanishing gradients problem [23]. A residual neural network is described by the equations,

$$y = F(x) + x \tag{4}$$

where $F(x)$ is a combination of matrix multiplication and activation functions. The residual blocks are stacked together to form the network. When a residual block is of the type $F : \mathbb{R}^n \to \mathbb{R}^m$, the $F(x) + x$ is undefined and a projection is used. With a projection the residual block is defined as,

$$y = F(x) + P(x) \tag{5}$$

where

$$P : \mathbb{R}^n \to \mathbb{R}^m \tag{6}$$

$$F : \mathbb{R}^n \to \mathbb{R}^m \tag{7}$$

2.1.5 Long-Short-Term Memory

The issues with the gradient descent in RNNs lead to the development of an architecture known as long-short-term memory (LSTM) which introduced gates and a constant error carousel into the RNN unit while still retaining the recurrent nature of the unit [22]. LSTM is more computationally intensive when compared to RNN and unlike the MLP or CNN is not parallelisable, however it performs best out of the models described thus far because of its gated design.

The developments from RNN introduced in LSTM are the input gate i, forget gate f, memory gate g, and the output gate o. [22]. The gates and states are calculated at time step t as follows,

$$f_t = \sigma(W_f \cdot [h_{t-1} \oplus x_t] + B_f) \tag{8}$$

$$i_t = \sigma(W_i \cdot [h_{t-1} \oplus x_t] + B_i) \tag{9}$$

$$o_t = \sigma(W_o \cdot [h_{t-1} \oplus x_t] + B_o) \tag{10}$$

$$g_t = \tanh(W_c \cdot [h_{t-1} \oplus x_t] + B_c) \tag{11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{12}$$

$$h_t = o_t \odot \tanh(c_t) \tag{13}$$

where W denotes a weight matrix, $\oplus$ denotes vector concatanation, $\sigma(a)$ denotes the sigmoid function, $\tanh(a)$ denotes the hyperbolic tangent, B a bias matrix, x the input, h the hidden state, and c the cell state. LSTM does not suffer from vanishing or exploding gradients like vanilla RNNs do because the constant error carousel [22] allows for information to flow through the cell states each iteration with only linear transformations as seen in figure 4 [24].

LSTMs were widely used for most sequential machine learning tasks [25] until the introduction of the transformer architecture [26]. LSTM architectures had three major

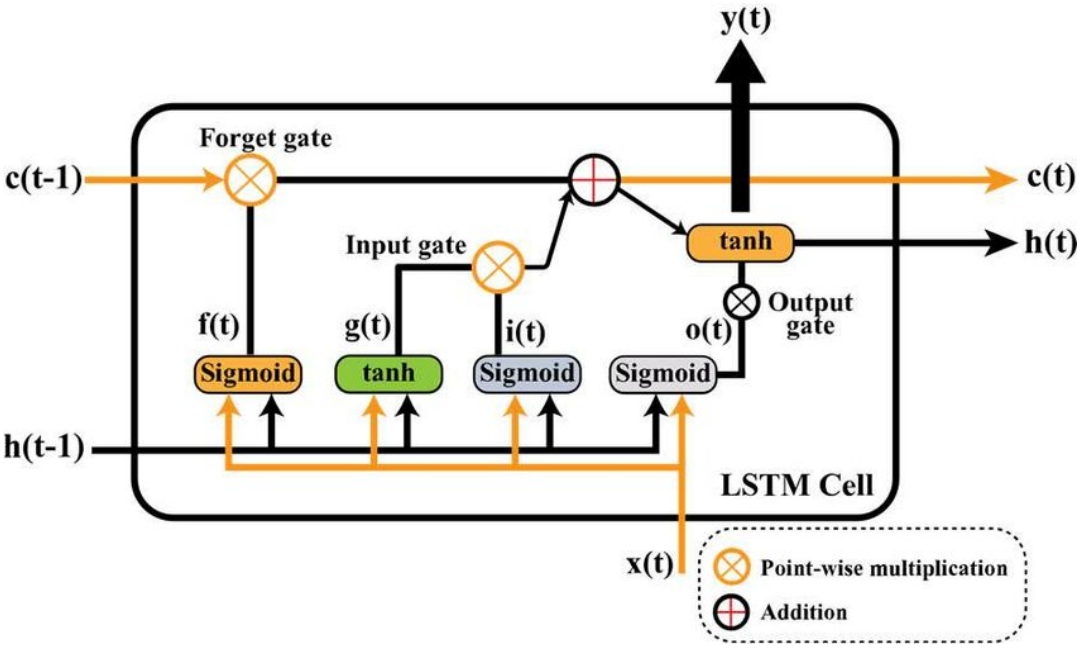


Figure 4: LSTM cell

drawbacks which made the transformer replace it so quickly. The three being: inability to revise storage decisions; limited storage capacity; and lack of parallelisability [25].

To overome the drawbacks in the standard LSTM architecture, the xLSTM architecture was developed [25] in 2024. The xLSTM or extended long-short-term memory is a combination of two LSTM variants, sLSTM and mLSTM. sLSTM introduces exponential gates combined with normalisation and stabilisation to the LSTM architecture allowing for the model to revise storage decisions. the sLSTM forward pass is described by the following set of equations,

$$f_t = \psi(W_f \cdot [h_{t-1} \oplus x_t] + B_f) \tag{14}$$

$$i_t = exp(W_i \cdot [h_{t-1} \oplus x_t] + B_i) \tag{15}$$

$$o_t = \sigma(W_o \cdot [h_{t-1} \oplus x_t] + B_o) \tag{16}$$

$$g_t = \tanh(W_c \cdot [h_{t-1} \oplus x_t] + B_c) \tag{17}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{18}$$

$$n_t = f_t \odot n_{t-1} + i_t \tag{19}$$

$$h_t = o_t \odot (c_t \odot n_t^{-1}) \tag{20}$$

where $\sigma(a)$ denotes the sigmoid function, $\tanh(a)$ denotes the hyperbolic tangent, $\psi(a)$ denotes either the exponential function or the sigmoid function, the symbols found in equations 8 to 13 denote the same data here as in LSTM, and the addition n denotes the new normaliser gate [25]. Due to the exponential gates sLSTM has to be stabilised with a simple mechanism to prevent the exponential gates from diverging to

infinity. The mechanism uses a m_t state and incorporates that into the input and forget gates. This stabilisation mechanism is described by equations 21 to 23 [25].,

$$m_t = max(log(f_t) + m_{t-1}, log(i_t)) \tag{21}$$

$$f'_t = exp(log(f_t) + m_{t-1} - m_t) \tag{22}$$

$$i'_t = exp(log(i_t) - m_t) \tag{23}$$

Beck et al. [25] showed that the incorporation of the stabilisation mechanism does not change either the numerical values of the function nor the derivatives through f_t and i_t.

The other addition to the xLSTM architecture is the mLSTM. The mLSTM introduces an improvement to the LSTM storage cabability by changing the memory cell scalar $c \in \mathbb{R}$ into a matrix $C \in \mathbb{R}^{d \times d}$ so the retrieval uses a matrix multiplication and uses a key vector $k \in \mathbb{R}^d$ to retrieve the memory as in Bidirectional Associative Memory [25]. For a key value pair the following equation is used [25],

$$C_t = C_{t-1} + v_t k_t^T \tag{24}$$

Rewriting the LSTM equations to incorporate these changes gives the forward pass of the mLSTM block as [25],

$$C_t = f_t C_{t-1} + i_t v_t k_t^T \tag{25}$$

$$n_t = f_t n_{t-1} + i_t k_t \tag{26}$$

$$h_t = o_t \odot \frac{C_t q_t}{max\{|n_t^T q_t|, 1\}} \tag{27}$$

$$q_t = W_q x_t + B_q \tag{28}$$

$$k_t = \frac{1}{\sqrt{d}} W_k x_t + b_k \tag{29}$$

$$v_t = W_v x_t + b_v \tag{30}$$

$$i_t = exp(W_i x_t + B_i) \tag{31}$$

$$f_t = \psi(W_f x_t + B_f) \tag{32}$$

$$o_t = \sigma(W_o x_t + B_o) \tag{33}$$

For i_t and f_t the same stabilisation technique as is used with sLSTM can be used [25]. Unlike in the LSTM or sLSTM blocks the mLSTM does not have memory mixing. This absence of memory mixing allows for the mLSTM block to be parallelised as each gate is not dependant on the previous hidden state [25].

To form the xLSTM architecture the sLSTM and mLSTM blocks are stacked together residually. Where the sLSTM and mLSTM blocks contain the above blocks with additional upscaling and convolutions as shown in figure 5. [25]

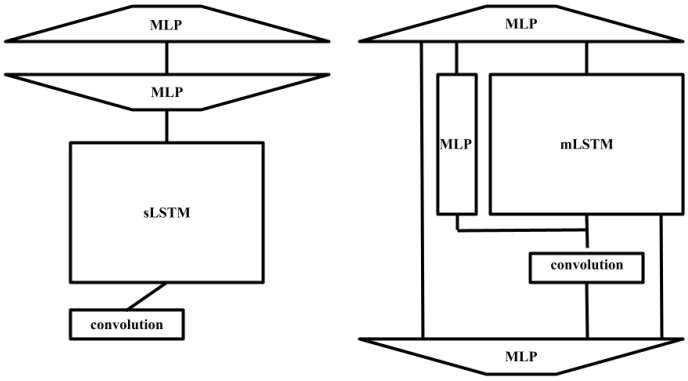


Figure 5: xLSTM building blocks: sLSTM on the left and mLSTM on the right, with the associated upscaling and convolutional layers

2.1.6 Transformer Network

The transformer architecture [26] was introduced in 2017 by researchers at Google. The transformer architecture aims to reduce the drawbacks of previous sequential models such as the LSTM and RNN. The transformer architecture introduces the attention mechanism which allows for the model to focus on different parts of the input at different times. In addition, the attention mechanism allows also for more parallellization du ring the training of the model, making training a transformer cheaper when compared to previous sequential models such as recurrent models.

The transformer architecture is what is called a encoder-decoder architecture [26]. The encoder converts the input sequence $\mathbf{x} \in \mathbb{N}^n$ into a continuous representation vector $\mathbf{z} \in \mathbb{R}^d$ where d is the dimension of the vector. The decoder then generates output sequence $\mathbf{y} \in \mathbb{N}^m$ from the continuous representation vector $\mathbf{z}$ [26].

The encoder and decoder are composed of a stack of N identical layers as shown in figure 6. Each layer having sublayers, the encoder having 2 and the decoder is having 3. In both the fist two sublayers are, multi-head self-attention mechnism and a fully connected feed-forward layer. For each of the two sublayers a residual connection is used along with a layer normalisation. In the decoder the third layer is a multi-head attention mechanism over the output of the encoder stack. [26]

The heart of the transformer architecture is the attention mechanism. The mechanism can be described as a vector function which takes in a set of key-value pairs and maps them to an output. The output is a weighted sum of the keys, with the weights for each value are the queries with a corresponding key calculated through a compatibility function. [26] The transformer architecture uses multi-head attention, in which the keys, values, and queries are first linearly projected to dimensions d_k, d_v and d_q respectively. After which the attention function is applied in parallel. Then

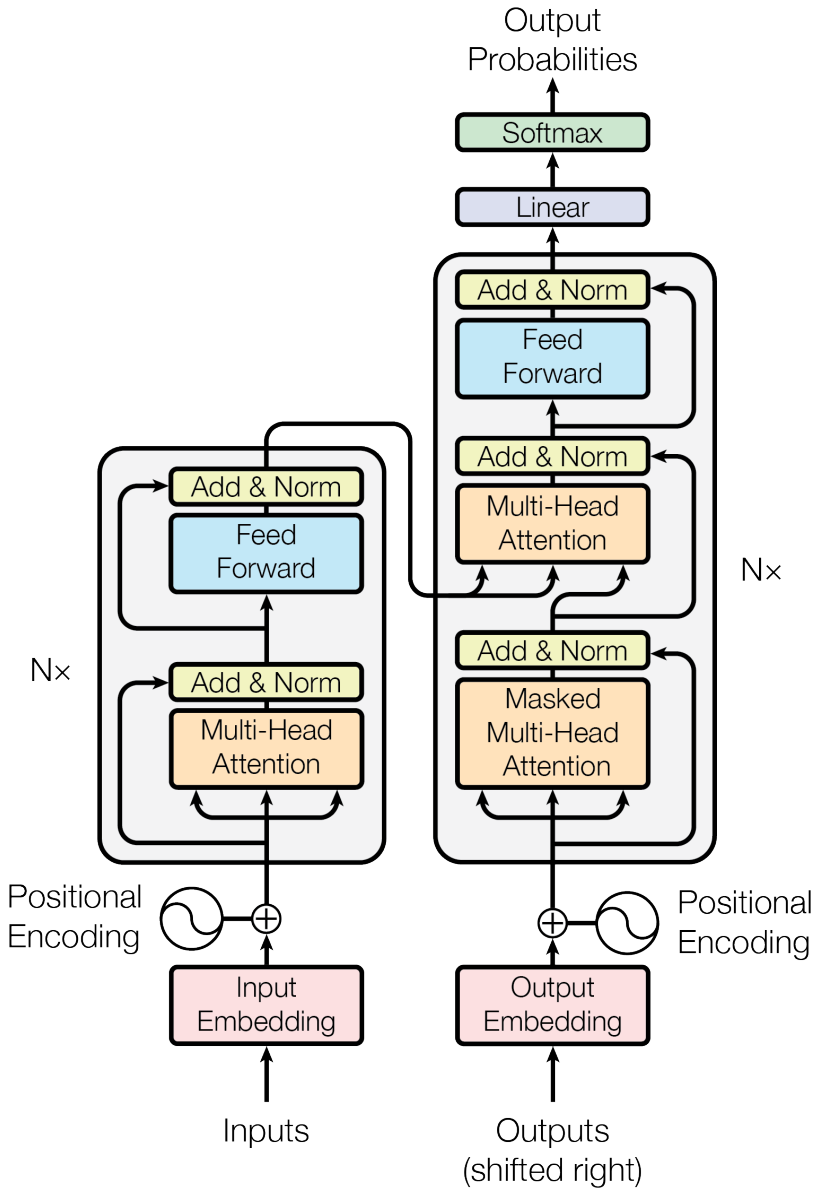


Figure 6: The transformer - model architecture [26]

these values are concatanated and projected to the wished dimension. Multi-head attention is described in equations 34-35.

$$MultiHead(Q, K, V) = concat(head_1, ..., head_h)W^O \tag{34}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{35}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$.

The other sublayers in the transformerr network are the feed-forward networks. [26] The feed-forward network is fully connected and uses the ReLU activation function giving us the equation,

$$FFN(x) = max(0, xW_1 + b1)W_2 + b_2 \tag{36}$$

In order to transform the input sequence into a numerical representation, a trained embedding layer is used to transform the input and output tokens into vectors of dimension d_{model}. [26] After the self-attention layers a final linear transformation and a softmax-function are used to calculate the final probabilities for the outputs [26].

The development of the transformer architecture has lead to the fast development of general-use generative large language models, generative pre-trainedtransformer and bidirectional encoder representations from transformers. These two models are widely used in applications such as ChatGPT, Llama, BERT, and Claude.

2.1.7 State Space Models

The newest promising emergence in the domain of sequential machine learning are state-space models. State-space models apply a method proposed in the 1960s for filtering and prediction problems [27] into sequential neural networks. In 2021, Gu et al. proposed a layer called Linear State-Space Layer [28], which generalized three major network types, recurrent, convolutional, and continuous-time.

The continuous-time case builds upon neural ordinary differential equations (ODE) introduced at NeurIPS [1] 2018. Neural ODEs build upon the similarities between residual neural networks and the euler's method for solving ODEs. [29]

A linear state-space layer (LSSL) maps a 1-dimansional function or sequence through an implicit state $x(t)$ by simulating a linear continuous-time state-space representation as shown in equations 37 and 38.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{37}$$

$$y(t) = Cx(t) + Du(t) \tag{38}$$

In this system A controls the evolution of the system, while B, C, and D are projection parameters [28].

The strength of LSSLs lies in their ability to be recurrent, convolutional, or continuous-time models. When LSSLs are used recurrently, a discrete time-step Δt is used to discretize the system into a linear recurrence. The equations 37- 38 are known to be representable as continuous convolutions [28]. When operating with a discrete model the layer can be efficiently parallelized during training due to its convolutional nature. The model can be viewed finally also as a continuous-time model as the equations 37-38 in themselves ordianry differential equations (ODE), and hence LSSLs can perform exceptionally with continuous-time problems and missing data points [28].

As LSSLs are a generalisation of RNN and CNN architectures, all 1-D convolutional kernels can be approximated by LSSLs and some RNN architectural properties such as gating-mechanisms are closely related to the discretized version and are deriveable from ODEs.

When discretizing the the system the matrices A and B are transformed from continuous representations using some discretization functions F and G into discrete representations $\overline{A}$ and $\overline{B}$ yielding the discretized version of the system.

[1] Conference on Neural Information Processing Systems

$$x_t = \overline{A}x_{t-1} + \overline{B}u_t \tag{39}$$

$$y_t = Cx_t + Du_t \tag{40}$$

The discrete form of an LSSL is also its recurrent form, while the convolutional variant can be calculated as shown in 41-44. When $x_{-1} = 0$ equations 39-39 yield,

$$y_k = C\overline{A}^k\overline{B}u_0 + C\overline{A}^{k-1}\overline{B}u_1 + ... + C\overline{AB}u_{k-1} + C\overline{B}u_k + Du_k \tag{41}$$

giving us

$$y_k = \sum_{i=0}^{k}(C\overline{A}^{k-i}\overline{B}u_i) + Du_k \tag{42}$$

This can be simplified to a non-circular convolution,

$$y = \mathcal{K}_{\mathcal{L}}(\overline{A}, \overline{B}, C) * u + Du \tag{43}$$

where

$$\mathcal{K}_{\mathcal{L}}(\overline{A}, \overline{B}, C) = (CA^iB)_{i\in[L]} \in \mathbb{R}^L = (CB, CAB, ..., CA^{L-1}B) \tag{44}$$

Hence the LSSL can be parallelized for a given sequential input, that is not recurrent, and the output $y \in \mathbb{R}^{H\times L}$ can be computed at once, reducing the computational complexity.

After the proposal of LSSL Gu et Al. proposed structured state spaces (S4) [30]. The S4 parametrisation of state space models improves the efficiency of the computation for all three views of SSMs: the continuous (A, B, C), the recurrent $(\overline{A}, \overline{B}, \overline{C})$, and the convolutional $(\overline{K})$ representations.

The S4 parametrisation tackles the computational efficiency with an approach based on concepts from linear algebra such as diagonilisation and conjugation. The bottleneck of computing discrete SSMs is the large amount of repeated matrix multiplication by $\overline{A}$. The approach can be applied to well-conditioned HiPPO matrices [31] which can be decomposed to Normal Plus Low-Rank (NPLR) [30].

The approach utilises three techniques to streamline the calculation of the convolution kernel. For the first technique the truncated generating function at the roots of unity is evaluated for $\overline{K}$ instead of computing it directly. After this $\overline{K}$ can be found by applying an inverse fast fourier transform [30]. The generating function relates to the matrix resolvent and involves a matrix inverse instead of power and thus the Woodbury identity reducing $\overline{A}$ to the diagonal case [30]. The third technique equates the diagonal matrix case to the computation of a cauchy kernel $\frac{1}{w_i-\zeta_k}$, which is a well-studied problem with stable linear algorithms [32, 33].

2.1.8 Selective State Spaces

The S4 model is defined by four parameters (Δ, A, B, C) in the context of sequence modeling. The parameters define a sequence-to-sequence transformation through an implicit latent state $h(t) \in \mathbb{R}^N$ descirbed by equations 45 - 52

$$h'(t) = Ah(t) + Bh(t) \tag{45}$$

$$y(t) = Ch(t) \tag{46}$$

$$h_t = \overline{A}h_{t-1} + \overline{B}h_{t-1} \tag{47}$$

$$y_t = Ch_t \tag{48}$$

$$\overline{K} = \{C\overline{A}^i\overline{B} | i = 0, i = k\} \tag{49}$$

$$y = x * \overline{K} \tag{50}$$

$$\overline{A} = f_A(\Delta, A) \tag{51}$$

$$\overline{B} = f_B(\Delta, A, B) \tag{52}$$

S4 model is linearly time invariant because (Δ, A, B, C) are fixed for all time steps and since (Δ, A, B) are invariant of time thus also $(\overline{A}, \overline{B})$ are also time invariant. This invariance reduces the models ability to contextualise the inputs and select which inputs to emphasise. To overcome this issue the Mamba architecture which uses Selective State Spaces (S6) was introduced [34]. The S6 model improves upon the S4 architecture by proposing a selection mechanism.

The proposed selection mechanism alters the equations in such a way that makes (Δ, B, C) functions of the input at each timestep. This change in the model leads to the model becoming time-varying instead of time-invariant. Due to this change also $(\overline{A}, \overline{B})$ are also dependant on the input and thus the S6 model can only be recursive as the convolution simplification of the S4 model does not work anymore. Despite this the S6 model can still be optimised through paralellisation with an efficient parallel scan algorithm.

The Mamba architecture [34] was further developed by bridging the connections between SSMs, structured matrices, and attention to establish state space duality (SSD) [35] for the Mamba 2 architecture. The paper more specifically uses the similarities between structured state space models and structured attention.

2.2 Clinical Coding

Clinical coding is the process of translating diagnoses, procedures, treatments, and other clinical information found in a patient's record into standardised alphanumeric codes. The codes belong to some controlled vocabulary of codes and the vocabulary can be general or for some specific purpose. Example of a general vocabulary is the International Classification of Diseases [6].

The currently most used clinical code vocabulary the ICD-10 constitutes over 120'000 codes [36]. This number of codes is a huge number of alphanumeric codes for humans to learn and use without machine aid. In Finland, the ICD-10 vocabulary has been in use since 1996 [37] with requirement from the European Union to switch to the newer ICD-11 by 2029. [38, 39] The ICD-11 has been designed to be fully digital and it aids in the transition of healthcare data to the 21st century. [40]

For the research into clinical coding different datasets have been created with MIMIC-III [41] being the most used dataset for research into automated clinical coding. MIMIC-III uses the ICD-9 [42] codec which was replaced in much of the western world by ICD-10. Although the MIMIC-III is the more used version still, a new version of the MIMIC database, MIMIC-IV, was published in 2023. [43] It improves upon MIMIC-III by including both the ICD-9 and the ICD-10 codecs.

2.3 Research of Language Models in Clinical Coding

The use of machine learning in clinical coding has been researched increasingly in the past years with the developments of AI. Various different models have been used thus far in the automation of clinical coding. The results have varied between models and studies.

CNNs have been used multiple times with one study achieving: precisions between 0.48 and 0.69; recalls between 0.78 and 0.91; and F-scores between 0.61 and 0.78 [44] while another CNN achieving: precision at around 0.5; recall around 0.37; and F-score of approximately 0.4 [8] A deep recurrent convolutional neural network approach achieved F1-micro scores ranging from 0.74 to 0.85. [9]

In more recent studies tranfromer models have been used. A study using transformer architecture achieved an average F1-score of 0.82. [45] Another transformer model achieved better accuracy for clinical coding than humans. [46]

3 Discussion

In recent years, there has been an increase in the application of LLMs in healthcare. Despite this the applications in automated clinical coding have remained low in numbers. The newest research dives into transformer models with RNNs and CNNs being still the most researched models in automated clinical coding. The newer state space models achieve better with increasing context sizes and significant memory improvements over trandsformers, RNNs and CNNs. This may limit the understanding of what LLMs can do with structured healthcare data.

One significant reason for the slow adoption of newer architectures in the context of healthcare data, is the lack of openly available high quality training sets. LLMs often require billions of tokens to train a high performing model, however well curated healthcare datasets are few in number. On top of which, healthcare data such as doctor's notes, anamnesis, details on medications, and discharge summaries are also highly protected by privacy laws and regulations such as GDPR and HIPAA. Even with de-identified datasets, regulatory bodies often impose restrictions on the use and distribution of a dataset containing real patient data.

After overcoming the problem with the scarcity of a data set the next problem is the linguistic variety that can be found in doctor's transcriptions as doctors often use arbitrary abbreviations, specialised language, and very general language all mixed together creating a semantic challenge. While the increase in number of abbreviations, symbols, and specialised words can be picked up by large language models, it is done

so at a high computational cost. The language of the doctor's notes also affects the automation of clinical coding as the transcripts would either have to first be translated into a lingua franca and then classified or the large language model would have to be trained in all languages it will be used in. This problem is emphasized in bilingual and multilingual countries such as Finland and Switzerland. In multilingual countries the national healthcare system operates in multiple languages and thus an LLM application in Switzerland for example would have to function in German, French and Italian. This either increases the required training data for the application or the need for machine translation before coding. Machine translation is challenging with highly specialised texts and training and running an LLM that works equally well in two or more languages is costly.

With these challenges, the first tackleable problem is the creation of high quality data sets in order that an LLM even could theoretically be trained. The second tackleable problem would be the lack of research into newer machine learning models for clinical coding applications. The final problem to be tackled would be the linguistic issue especially for multilingual countries.

4 Conclusion

In this thesis, we have explored the evolution of neural network architectures that form the foundation of large language models. Starting from basic multilayer perceptrons, we examined their limitations in scalability and sequence modeling, which led to the development of more advanced architectures such as convolutional neural networks and recurrent neural networks. The progression continued through long short-term memory networks and their enhanced variants like xLSTM, addressing critical issues such as vanishing gradients, limited memory, and lack of parallelism. This architectural evolution culminated in the introduction of the transformer, which revolutionized NLP by enabling efficient parallel processing and flexible attention mechanisms. More recently, state-space models , including the S4 and S6 families, have emerged as promising alternatives, offering efficient, scalable handling of long sequences while blending properties of RNNs and CNNs.

These improvements in language modeling are positive winds for the future automation of clinical coding applications as longer context frame, more efficient training, and increasingly accurate models are what are needed. However despite a promising direction, more research is needed into the automation of clinical coding with the help of modern large language models.

References

[1] OpenAI, "ChatGPT (GPT-4o)," https://chatgpt.com/share/503cc819-0f8b-4e32-91e4-f80015e0d5cb, 2024, accessed: Jun. 15, 2024.

[2] R. Perrault and J. Clark, "Artificial intelligence index report 2024," Stanford University Human-Centered Artificial Intelligence, Tech. Rep., 2024.

[3] D. Brin, V. Sorin, A. Vaid, A. Soroush, B. Glicksberg, A. Charney, G. Nadkarni, and E. Klang, "Comparing chatgpt and gpt-4 performance in usmle soft skill assessments. sci rep 13, 16492," 2023.

[4] O. E. Karpov, E. N. Pitsik, S. A. Kurkin, V. A. Maksimenko, A. V. Gusev, N. N. Shusharina, and A. E. Hramov, "Analysis of publication activity and research trends in the field of ai medical applications: Network approach," *International Journal of Environmental Research and Public Health*, vol. 20, no. 7, p. 5335, 2023.

[5] M. Tian, Z. Shen, X. Wu, K. Wei, and Y. Liu, "The application of artificial intelligence in medical diagnostics: A new frontier," *Academic Journal of Science and Technology*, vol. 8, no. 2, pp. 57–61, 2023.

[6] World Health Organization, *International Statistical Classification of Diseases and Related Health Problems, 10th Revision (ICD-10), Fifth Edition*. Geneva: World Health Organization, 2016, fifth consolidated print edition. [Online]. Available: https://iris.who.int/handle/10665/246208

[7] M. Subotin and A. Davis, "A system for predicting icd-10-pcs codes from electronic health records," in *Proceedings of bionlp 2014*, 2014, pp. 59–67.

[8] M. Zeng, M. Li, Z. Fei, Y. Yu, Y. Pan, and J. Wang, "Automatic icd-9 coding via deep transfer learning," *Neurocomputing*, vol. 324, pp. 43–50, 2019.

[9] S. R. Bhutto, Y. Wu, M. Zeng, A. W. Dogar, K. Ullah, and M. Li, "Drcnntle: A deep recurrent convolutional neural network with transfer learning through pre-trained embeddings for automated icd coding," *Methods*, vol. 205, pp. 97–105, 2022.

[10] A. Chraibi, D. Delerue, J. Taillard, I. Chaib Draa, R. Beuscart, and A. Hansske, "A deep learning framework for automated icd-10 coding," in *Public Health and Informatics*. IOS Press, 2021, pp. 347–351.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[12] R. C. Ghosh, P. Shailendra, and G. B. Singh, "Large language model in various fields: Opportunities, challenges and risks," in *NIELIT's International Conference on Communication, Electronics and Digital Technologies*. Springer, 2024, pp. 587–596.

[13] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[14] J. He, L. Li, J. Xu, and C. Zheng, "Relu deep neural networks and linear finite elements," *arXiv preprint arXiv:1807.03973*, 2018.

[15] H. Mohamed, A. Negm, M. Zahran, and O. C. Saavedra, "Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake," in *International water technology conference*, 2015, pp. 12–14.

[16] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021.

[17] B. Brinkworth, "Examples of Changes in Different Bible Versions," http://www.openthoumineeyes.com, accessed: Sep. 8, 2024.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Pascal*. Cambridge University Press, 1989.

[19] P.-H. Kuo and C.-J. Huang, "A green energy application in energy management systems by an artificial intelligence-based solar radiation forecasting model," *Energies*, vol. 11, no. 4, p. 819, 2018.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986," *Biometrika*, vol. 71, no. 599-607, p. 6, 1986.

[21] S. Li, "Text classification based on machine learning methods," Master's thesis, 2019.

[22] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] M. B. A. Tamal, M. A. Alam, M. N. Sharker, and M. I. Sazib, "Forecasting of solar photovoltaic output energy using lstm machine learning algorithm," in *2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI)*. IEEE, 2022, pp. 1–6.

[25] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xlstm: Extended long short-term memory," *arXiv preprint arXiv:2405.04517*, 2024.

[26] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[27] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[28] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems*, vol. 34, pp. 572–585, 2021.

[29] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[30] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[31] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in neural information processing systems*, vol. 33, pp. 1474–1487, 2020.

[32] V. Pan, "Fast approximate computations with cauchy matrices and polynomials," *Mathematics of Computation*, vol. 86, no. 308, pp. 2799–2826, 2017.

[33] V. Y. Pan, "Transformations of matrix structures work again," *Linear Algebra and Its Applications*, vol. 465, pp. 107–138, 2015.

[34] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[35] T. Dao and A. Gu, "Transformers are ssms: Generalized models and efficient algorithms through structured state space duality," *arXiv preprint arXiv:2405.21060*, 2024.

[36] K. J. O'malley, K. F. Cook, M. D. Price, K. R. Wildes, J. F. Hurdle, and C. M. Ashton, "Measuring diagnoses: ICD code accuracy," *Health services research*, vol. 40, no. 5p2, pp. 1620–1639, 2005.

[37] Sosiaali- ja terveysministeriö, "WHO:n ICD-10 -tautiluokituksen ja pohjoismaisen leikkausluokituksen käyttöönottoa koskeva määräys," *Sosiaali- ja terveysministeriön määräyskokoelma*, no. D:13/332/95, 1995.

[38] E. Union, "Regulation (eu) 2025/327 of the european parliament and of the council of 11 february 2025 on the european health data space and amending directive 2011/24/eu and regulation (eu) 2024/2847 (text with eea relevance)," *OJ*, vol. L 327, 2025.

[39] World Health Organization. (2022) International classification of diseases for mortality and morbidity statistics (icd-11 mms). World Health Organization. Eleventh Revision; official release 11 Feb 2022, effective 1 Jan 2022. [Online]. Available: https://icd.who.int/browse/latest-release/mms/en

[40] J. E. Harrison, S. Weber, R. Jakob, and C. G. Chute, "Icd-11: an international classification of diseases for the twenty-first century," *BMC medical informatics and decision making*, vol. 21, pp. 1–10, 2021.

[41] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[42] V. N. Slee, "The international classification of diseases: ninth revision (icd-9)," pp. 424–426, 1978.

[43] A. E. Johnson, L. Bulgarelli, L. Shen, A. Gayles, A. Shammout, S. Horng, T. J. Pollard, S. Hao, B. Moody, B. Gow *et al.*, "Mimic-iv, a freely accessible electronic health record dataset," *Scientific data*, vol. 10, no. 1, p. 1, 2023.

[44] J. H. B. Masud, C. Shun, C.-C. Kuo, M. M. Islam, C.-Y. Yeh, H.-C. Yang, and M.-C. Lin, "Deep-adca: development and validation of deep learning model for automated diagnosis code assignment using clinical notes in electronic medical records," *Journal of Personalized Medicine*, vol. 12, no. 5, p. 707, 2022.

[45] N. Thapa, "Sequence-to-sequence learning for predicting procedure codes from unstructured clinical text," *Transactions on Machine Learning, Artificial Intelligence, and Advanced Intelligent Systems*, vol. 15, no. 1, pp. 1–12, 2025.

[46] Y. Yin, Y. Shao, P. Ma, Q. Zeng-Treitler, and S. J. Nelson, "Machine-learned codes from ehr data predict hard outcomes better than human-assigned icd codes," *Machine learning and knowledge extraction*, vol. 7, no. 2, p. 36, 2025.