Deep Transformer Neural Networks with Stochastic Competition

Andreas Voskou

Doctoral Dissertation

Cyprus University of Technology

Faculty of Engineering and Technology

Department of Electrical Engineering, Computer Engineering and Informatics

Limassol, April 2024

Approval Form

Doctoral Dissertation

Deep Transformer Neural Networks with Stochastic Competition

Presented by

Andreas Voskou

Supervisor: Dr. Sotirios Chatzis, Associate Professor, Cyprus University of Technology

Signature _____

Member of the committee: Dr. Michael Sirivianos, Associate Professor, Cyprus University of Technology

Signature _____

Member of the committee: Prof. George Bebis, Professor, University of Nevada, Reno

Signature _____

Cyprus University of Technology

Limassol, April 2024

Copyrights

The approval of the dissertation by the Department of Electrical Engineering, Computer Engineering and Informatics does not imply necessarily the approval by the Department of the views of the writer.

Acknowledgments

First and foremost, I extend my sincere gratitude to my supervisor, Prof. Sotirios Chatzis, for providing me with this invaluable opportunity. His trust, mentorship, and willingness to share his deep knowledge have been pivotal to my growth.

Additionally, my gratitude extends to numerous individuals and organizations that have supported me throughout this journey. I am particularly thankful to the entire team at the Statistical Machine Learning Lab and all the external partners for their collaboration and mutual assistance. A special thanks to Harris Partaourides, whose invaluable advice and guidance during the initial months of my doctoral studies were instrumental. I also wish to express my appreciation to my colleagues, Chariton and Maria, for sharing their experiences and expertise providing assistance in several technical matters and beyond. This endeavor would not have been possible without the generous support and financial assistance from Tickmill, Boltzmann Research, and the EU research and innovation programs AiD (Horizon 2020) and CatSL (Erasmus+).

This thesis is dedicated to those who have unconditionally supported me over the years.

Abstract

Transformers have become one of the most successful architectures in deep learning, experiencing a steady rise in popularity. These advanced networks have revolutionized the field of Natural Language Processing (NLP) and are extending their influence into new domains within artificial intelligence and beyond. The recent rise of large language models, fundamentally reliant on Transformer architectures, highlights their effectiveness and underscores their transformative impact.

This thesis delves into exploring further capabilities of this deep learning framework by incorporating stochastic methodologies as an essential component of Transformer networks. Our primary focus is on leveraging stochastic competition techniques, proven to be highly advantageous in various contexts, as the cornerstone for developing high-performing models. Instead of focusing on the extensively researched application areas such as NLP, our research pivots to exploring two distinct and significantly different fields: i) Sign Language Translation and ii) Tabular Data Modeling.

We begin with a short introduction to the concept of Deep Learning and its basic techniques. We outline the foundational concepts of neural networks, the core principles of training procedures, and some simple and widely-used modules. Furthermore, we address common challenges, such as overfitting, and the typical strategies for dealing with them. In the second chapter, we progress to Bayesian Neural Networks. Here, we introduce Bayesian statistics and its application in re-approximating classical neural networks (NNs). We elaborate on the most significant methods of this approach and the robust properties they inject into neural networks, thereby essentially defining and introducing Stochastic Neural Networks, as they are approached in this thesis. Following this, we explore modern methods in deep learning for processing sequential structures and Natural Language, aiming to establish the necessary background leading to the introduction of Transformer networks. We then delve into the detailed formulation of this influential paradigm, including its original format and subsequent variants.

Advancing towards the main areas of contribution, the fourth chapter is dedicated to Sign Language and, more specifically, automatic Sign Language Translation. We

aim to build the required background on the applications of neural networks in the recognition and translation of Sign Language. Beyond its scientific intrigue, the task of SLT is of considerable social importance, bridging communication gaps between the deaf and the hearing. Herein, we propose a novel SLT methodology that employs a specialized Transformer architecture equipped with stochastic modalities, including: (i) local winner-takes-all (LWTA) layers with stochastic winner sampling, as opposed to conventional ReLU layers, (ii) stochastic weights with posterior distributions estimated via variational inference, and (iii) a weight compression technique at inference time leveraging estimated posterior variance for substantial, nearly lossless compression. The proposed model demonstrates state-of-the-art results on the Phoenix14T dataset, the standard benchmark for such models. Moving beyond this innovation, we aim for real-world applicability. Existing SLT methods either lack translation capability or are trained and evaluated on datasets with limited vocabulary and real-world applicability. An example is the aforementioned Phoenix2014T benchmark dataset, restricted to weather forecasts in German Sign Language. Addressing this gap, we introduce a new collection of 29,653 Greek Sign Language video translation pairs based on the official Greek Elementary School syllabus and covering a broad range of subjects. Utilizing this novel dataset, we develop a new variant of our model. Our findings highlight our method's potential, striking a favorable balance between predictive power, usability and real-world applicability.

In the fifth chapter, we expand our research to include modeling of tabular data. This task is of significant interest, as such data forms frequently appear across numerous scientific and technical fields. Interestingly, despite their popularity, importance, and seeming simplicity, these data formats have been historically underexplored from a deep learning perspective, though they have started garnering increasing attention in recent years. By employing a Stochastic Competition Hybrid Transformer, we achieved exceptional results across multiple well-known tabular datasets, surpassing previous deep learning networks and the state-of-the-art Gradient Boosting methods that have dominated this area. These findings underscore the effectiveness of Stochastic Competition Transformers in a broad and diverse range of domains.

Οι Transformers έχουν αναδειχθεί ως ένα από τα πιο επιτυχημένα παραδείγματα αρχιτεκτονικής στην βαθιά μάθηση, με μεγάλη και σταθερά αυξανόμενη δημοτικότητα. Η συγκεκριμένη οικογένεια νευρωνικών δικτύων έχει, στο πρόσφατο παρελθόν, φέρει επανάσταση στον τομέα της Επεξεργασίας Φυσικής Γλώσσας (NLP) και επεκτείνεται με ραγδαίους ρυθμούς και σε άλλους τομείς. Η πρόσφατη άνοδος των μεγάλων γλωσσικών μοντέλων (LLMs), που βασίζονται στην μεγάλη πλεοψηφία τους σε Transformer, αποτελεί ίσως το ποιό ηχηρό παράδείγμα της κυριαρχίας τους στη σύγχρονη εποχή της τεχνιτής νοημοσύνης.

Αυτή η διατριβή εξετάζει τις δυνατότητες και τις προεκτάσεις αυτού του πλαισίου βαθιάς μάθησης συνδιαστικά με την ενσωμάτωση στοχαστικών μεθοδολογιών στα θεμελιώδη δομικά μέρη των δικτύων Transformer. Πιο συγκεκριμένα, εστιάζουμε κυρίως στην αξιοποίηση τεχνικών στοχαστικού ανταγωνισμού, οι οποίες έχουν αποδειχθεί ιδιαίτερα πλεονεκτικές σε διάφορα πλαίσια βελετιώνοντας τις απόδοσεις αντίστοιχων ντετερμινιστικών μοντέλων. Η έρευνά μας στρέφεται πέραν των εκτενώς μελετημένων περιοχων εφαρμογής (NLP, κλπ), προς την μελέτη δύο εναλλακτικών και ανόμιων μεταξύ τους πεδίων: i) την Μετάφραση Νοηματικής Γλώσσας και ii) την Μοντελοποίηση Δεδομένων Πινάκων, σκοπεύοντας στην γενίκευση των συμπερασμάτων μας.

Η διατριβή ξεκινά με μια επιγραμματική εισαγωγή στις έννοιες της Βαθιάς Μάθησης και τις βασικότερς τεχνικές, προσφέροντας εν συντομία στον αναγνώστη το απαραίτητο υπόβαθρο. Ομοίως στο δεύτερο κεφάλαιο, προχωρούμε στην εισαγωγή των Μπαεζιανών νευρωνικών δικτύων παραθέτοντας βασικές έννοιες Μπαεζιανής στατιστικής και την εφαρμογή της στην επαναπροσέγγιση των κλασικών νευρωνικών δικτύων, εισάγοντας έτσι τα στοχαστικά νευρωνικά δίκτυα όπως αυτά προσγγίζονται στην παρούσα διατριβή. Ακολούθως, στο τρίτο κεφάλαιο παρουσιάζουμε τις σύγχρονες μεθόδους βαθιάς μάθηση για την επεξεργασία σειριακών δομών και Φυσικής Γλώσσας, αποσκοπόντας πέραν από την κάλυψη του απαραίτητου υπόβαθρου στην ομαλή εισαγωγή των δικτύων Transformer, όντας η πλέον προηγμένη προσέγγιση στον χώρο. Επιπλέον, περιγράφουμε λεπτομερώς την αυθεντική τους μορφή, παραλλαγές και επεκτάσεις τους.

Προχωρώντας προς τις κύριες περιοχές συνεισφοράς, το τέταρτο κεφάλαιο είναι αφιερωμένο στη Νοηματική Γλώσσα και, πιο συγκεκριμένα, στην αυτόματη μετάφραση Νοηματικής

Γλώσσας με την χρήση σύγχρονης μηχανικής μάθησης, γνωστή ως SLT (Sign Language Translation). Πέρα από το τεχνικό και επιστημονικό ενδιαφέρον, το εν λόγω πρόβλημα έχει βαθύτατη κοινωνική επέκταση, γεφυρώνοντας κενά επικοινωνίας μεταξύ των κωφών και των ακουόντων. Αφού εισάγουμε τις απαραίτητες σχετικές έννοιες και την πρόσφατη βιβλιογραφία, προτείνουμε μια καινοτόμα μεθοδολογία που χρησιμοποιεί μια εξειδικευμένη αρχιτεκτονική Transformer εξοπλισμένη με 3 μορφές στοχαστικών διεργασιών, συγκεκριμένα: (i) Δίκτυα local winner-takes-all (LWTA) με στοχαστική επιλογή νικητή, αντι συμβατικών ντετερμινιστικών στοιχείων ReLU. (ii) Στοχαστικά βάρη μέσω της χρήσης της τεχνικής variational inference. (iii) Μια τεχνική συμπίεσης βαρών κατά τη διάρκεια της συμπερασματικής ανάλυσης, εκμεταλλευόμενοι την εκτιμώμενη μεταβλητότητα, για σημαντική και με ελάχιστες απώλειες, συμπίεση του δικτύου. Το προτεινόμενο μοντέλο επιδεικνύει κορυφαία αποτελέσματα στο σύνολο δεδομένων Phoenix14T, το βασικότερο μέτρο σύγκρισης τέτοιου τύπου μοντέλων.

Αποσκοπώντας σε πραγματικές εφαρμογές πέραν των θεωρητικών συγκρίσεων, αλλά και σε επέκταση στην υπομελετημένη ελληνική νοηματική γλώσσα, προχωρούμε στη χρήση μιας τροποποιημένης μορφής της μεθοδολογίας μας σε πιο ρεαλιστικές συνθήκες. Οι υπάρχουσες μέθοδοι SLT είτε υστερούν σοβαρά στην ικανότητα μετάφρασης, είτε εκπαιδεύονται και αξιολογούνται σε σύνολα δεδομένων με περιορισμένο λεξιλόγιο και θεματολογία, και ταυτόχρονα με αμφίβολη εφαρμοσιμότητα στον πραγματικό κόσμο. Αντιμετωπίζοντας αυτό το κενό, εισάγουμε μια νέα συλλογή αποτελούμενη από 29.653 βίντεο ελληνικής νοηματικής γλώσσας, συνοδευόμενα από τις αντίστοιχες μεταφράσεις στην ομιλούμενη ελληνική γλώσσα, όλα παρμένα από το επίσημο πρόγραμμα σπουδών των ελληνικών σχολείων δημοτικής εκπαίδευσης. Το σύνολο δεδομένων μας καλύπτει μια ευρεία γκάμα θεματικών ενοτήτων μέσω του συνδυασμού 6 διαφορετικών μαθημάτων. Τα αποτελέσματα της εκπαίδευσης της τροποποιημένης προτεινόμενης μεθοδολογίας στα νεοσυλλεγέντα δεδομένα υπογραμμίζουν τις δυνατότητες της μεθόδου μας, επιτυγχάνοντας μια ευνοϊκή ισορροπία μεταξύ μεταφραστικής ικανότητας, χρηστικότητας και εφαρμοσιμότητας στον πραγματικό κόσμο.

Στο πέμπτο κεφάλαιο, εξερευνούμε τις εφαρμογές των μοντέλων Transformer με στοιχεία στοχαστικού ανταγωνισμού σε δεδομένα πίνακα. Αυτός ο τομέας αποτελεί μια προέκταση

ιδιαίτερου ενδιαφέροντος λόγω της συχνής εμφάνισης αυτού του τύπου δεδομένων σε διάφορα επιστημονικά και τεχνικά πεδία. Όμως, παρά την σημαντικότητα και την φαινομενική δομική απλότητα τους, η χρήση βαθιάς μάθησης για τη μοντελοποίησή τους άρχισε να λαμβάνει την πρέπουσα προσοχή σχετικά πρόσφατα. Με την εφαρμογή των εν λόγω μοντέλων, πετύχαμε εξαιρετικά αποτελέσματα σε πολλά δημοφιλή σύνολα δεδομένων πινάκων, ξεπερνώντας τόσο προηγούμενα βαθιά δίκτυα όσο και μεθόδους Gradient Boosting που έχουν κυριαρχήσει στον τομέα αυτό τα τελευταία χρόνια. Αυτά τα αποτελέσματα επιβεβαιώνουν την αποτελεσματικότητα της προτεινόμενης οικογένειας μοντέλων σε διαφορετικούς τομείς εφαρμογών.

Contents

List of Tables

List of Figures

Publications

Andreas Voskou, Konstantinos Panousis, Dimitris Metaxas, and Sotirios Chatzis. "Stochastic Transformer Networks With Linear Competing Units: Application to end-to-end SL Translation". Published in: Proceedings of the International Conference on Computer Vision, 2021.

Andreas Voskou, Konstantinos Panousis, Harris Partaourides, Kyriakos Tolias, and Sotirios Chatzis. "A New Dataset for End-to-End Sign Language Translation: The Greek Elementary School Dataset". Published in: Proceedings of the International Conference on Computer Vision (ACVR), 2023.

Andreas Voskou, Charalampos Christoforou, and Sotirios Chatzis. " Transformers with Stochastic Competition for Tabular Data Modelling ". Under review for: European Conference on Machine Learning, 2024.

Joint Publications

Harris Partaourides, ***Andreas Voskou***, Dimitrios Kosmopoulos, Sotirios Chatzis, Dimitris N. Metaxas. "Variational bayesian sequence-to-sequence networks for memory-efficient sign language translation". Published in: Proceedings of the International Symposium on Visual Computing, 2020.

Chapter 1

Introduction to Deep Learning

Machine Learning is the field of Artificial Intelligence that employs mathematical modeling to inductively build reliable intelligence systems utilizing historical observations and data. At its core, Machine Learning is about learning a function $f : \mathcal{X} \to \mathcal{Y}$ that maps input values x sampled from an input class $\mathcal{X}$ to corresponding output values y from a paired output class $\mathcal{Y}$. Often, the function f is parameterized by a set of trainable variables $\boldsymbol{\theta}$, so that $y = f(\boldsymbol{x}|\boldsymbol{\theta})$. Here, "learning" refers to the process of finding the appropriate $\boldsymbol{\theta}$.

Deep Learning, a subset of Machine Learning, has dominated various sub-fields over the past decade, revolutionizing AI and numerous other sectors. It is based on Artificial Neural Networks, particularly those with large-scale architectures. Deep Learning operates through a series of geometric transformations, re-expressing data into latent, target-aware representative spaces. Deep models are unparalleled in processing raw, structured data with spatial and/or temporal dynamics, requiring minimal statistical preprocessing.

In this chapter, we offer a short overview of Artificial Neural Networks (ANNs) and Deep Learning, highlighting their fundamental principles, architectures, and operational mechanisms. We delve into a variety of ANN configurations and address prevalent challenges encountered in their application. Moreover, we summarize the core components of principal learning algorithms, specifically backpropagation and gradient descent, while also touching upon their notable variations. Concluding this section, we provide a brief discussion on the role of nonlinearity within neural networks and introduce several widely recognized activation functions.

1.1 Artificial Neural Networks

Artificial Neural Networks are biological inspired structures used for many machine learning applications. This computational model tries to mimic the way the human brain theoretically works in order to process logical tasks. The application of ANNs today are numerous and includes automation of core human processes like image, video and speech processing as well as complex systems like medical diagnosis.

The core unit behind ANNs is the Artificial Neuron a simple structure inspired from the biological neuron. The idea about this simple structure was firstly proposed by McCulloch and Pitts back in 1943 (Mcculloch &Pitts, 1943; Jain et al., 1996) . Artificial Neuron works by receiving one or more inputs and calculating a weighted sum of them. All the neuron connections between input and output are formed by the use of a weight matrix W, a bias vector $\mathbf{b}$ and an non-linear activation function σ. With more detail the output is calculated by multiplying the neurons of each layer with the corresponding weight matrix and using the sum of their product and the bias vector as argument to the activation function.

$$g(\mathbf{x}) = \sigma(W\boldsymbol{x} + \mathbf{b}) \tag{1.1}$$

Perceptron was the first learning model based on a single Artificial Neuron. This is about a binary classifier that maps a real or binary input vector $\boldsymbol{x}$ to a binary output y. If the weighted sum of the input is greater than 0 then the output value is y=1 otherwise y=0 1.2.

$$y(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

Although the idea was promising and effective for various tasks, it soon became apparent that it had limited recognition abilities. The perceptron learning algorithm can be trained to classify data if, and only if, the learning set is linearly separable. In cases where classes are not linearly separable, single neuron models prove ineffective, necessitating a more complex structure. Subsequently, it was discovered that combining multiple perceptrons results in significantly more powerful models.

A neural network is comprised of multiple interconnected artificial neurons. It resembles a weighted, directed graph consisting of several connected nodes arranged in more than two sequential layers. The first layer, known as the input layer, and the last layer, serving as the network's output or result, are essential. The sizes and structures of these two layers are determined by the nature of the input and the required outcome. The layers situated between the input and output, termed hidden layers, are optional. While there are no strict guidelines regarding the number and size of hidden layers, these factors critically influence the model's performance.

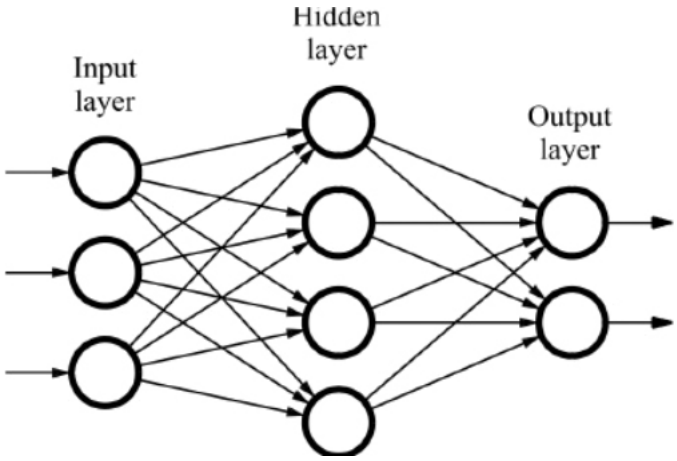

Figure 1.1: Simple Neural Network

Neural Networks (NNs) can be categorized into various types based on the coupling topology between neurons. The most basic form is the fully connected feed forward Neural Network also known as Multi Linear Perceptron (MLP). In this architecture, each neuron is connected to every neuron in both the preceding and succeeding layers. The signal flows seamlessly from the input layer to the output, influenced by all neurons in the hidden layers. Other types of networks incorporate more sophisticated functionalities. Nevertheless, the fundamental mechanism remains largely similar, relying on trainable weights and biases.

A characteristic example of a slightly more sophisticated network module is Convolutional Neural Networks (CNNs). These networks utilize local connections rather than fully connected layers, and are specifically tailored to process data with significant geometric attributes, spatial or temporal, across any dimension, though they are most frequently applied to 2-dimensional images. This process leverages a sliding window approach to

highlight features such as edges or textures within the input. Due to their local connectivity CNNs offer a computationally efficient approach and allow for particularly deep architecture. These structures are often combined with fully connected parts placed in the later stages for the final output, as well as other auxiliary modules such pooling that reduce the dimentionality of the hidden layers.

1.2 Backpropagation and Learning

Weights and biases are the aspect of learning of a Neural Network. At the beginning of the learning procedure, these two families of variables are randomly initialized. Subsequently, through small and stable steps, they progressively move towards the correct values. Because of their complex structure modern Deep Models are highly sensitive to the values of weights, where even minor changes can dramatically alter the final result.

The training process in a Deep Learning task is essentially a search for the weights and biases that best fit the examples in a given training dataset, as well as other similar but independent examples. This process typically relies on Backpropagation. Backpropagation is a method that involves running the result of the forward application of the model backwards, adjusting the parameters to enhance the quality of the results. To implement this procedure in a neural network, a loss function, denoted as L, is necessary.

The loss function in general, evaluates the deviation of the predicted results in comparison to the expected values. The loss must be continuous and differentiable in the domain of the results because of the nature of the learning algorithms which searches in the weight space for the values that minimize the loss function. In case of regression problems the typical Loss function is the Mean Square Error 1.3 and for classification a loss function based on cross entropy 1.4, usually with a sotfmax activation function.

$$L_2 = \frac{1}{N}\sum(y_{pre} - y_{exp})^2 \tag{1.3}$$

$$L_{CE} = -\frac{1}{N}\sum(y_{pre}\log(y_{exp})) \tag{1.4}$$

Gradient descent is a family of optimization algorithms extensively used in neural

network training. Vanilla gradient descent, or simply gradient descent, is the simplest training algorithm, searching for the minimum by making gradient-driven walks in the model's parameter space towards the optimal point. At each step, the new value of a weight is calculated by subtracting from its current value the product of the partial derivative of the loss with respect to the weight and a constant γ. The smaller γ is, the more accurate but slower the training algorithm becomes. The equation 1.9 shows the exact expression of an updating step for any weight $w \in \{\boldsymbol{w}\}$ with $\{\boldsymbol{w}\}$ the set of all the trainable parametes.

$$w \leftarrow w + \Delta w, \quad \Delta w = -\gamma \frac{\partial L(\mathcal{D}|\{\boldsymbol{w}\})}{\partial w}, \quad \forall w \in \{\boldsymbol{w}\} \tag{1.5}$$

For the output layer the gradient can be easily calculated by applying the chain rule.

$$\frac{\partial L}{\partial w_{ij}^N} = \frac{\partial z_i^N}{\partial w_{ij}^N} \frac{\partial a_i^N}{\partial z_i^N} \frac{\partial L}{\partial a_i^N} \tag{1.6}$$

where

- $z_i^N = \sum_j w_{ij}^N x_j^N$, the weighted sum of inputs to the neuron in the output layer.
- $a_i^N = \sigma(z_i^N)$, the activation of the neuron in the output layer.

For the layers N-1 and N-2 (out of the total N layer) and by using the chain rule again the gradient is given by the equations 1.7 and 1.8:

$$\frac{\partial L}{\partial w_{ij}^{N-1}} = \frac{\partial z_i^{N-1}}{\partial w_{ij}^{N-1}} \frac{\partial a_i^{N-1}}{\partial z_i^{N-1}} \sum \left(\frac{\partial z_j^N}{\partial a_j^{N-1}} \frac{\partial a_j^N}{\partial z_j^N} \frac{\partial L}{\partial a_j^N} \right) \tag{1.7}$$

$$\frac{\partial L}{\partial w_{ij}^{N-2}} = \frac{\partial z_i^{N-2}}{\partial w_{ij}^{N-2}} \frac{\partial a_i^{N-2}}{\partial z_i^{N-2}} \sum \left(\frac{\partial z_k^{N-1}}{\partial a_k^{N-2}} \frac{\partial a_k^{N-1}}{\partial z_k^{N-1}} \sum \left(\frac{\partial z_j^N}{\partial a_j^{N-1}} \frac{\partial a_j^N}{\partial z_j^N} \frac{\partial L}{\partial a_j^N} \right) \right) \tag{1.8}$$

For the rest layers the equations get more complicated but with the same way as before.

The ordinary gradient descent algorithm computes the gradient of the loss function using the whole dataset on each step. This way guarantee converge to the minimum (global or local) of the training data but has a few major drawbacks. Firstly it can be remarkably expensive especially in terms of memory usage. Modern DNNs can be enormous with hundred of millions of trainable parameters fact that makes this approach practically

impossible. Second using the whole dataset usually leads to great result on the training data but fails to generalise well on out of sample cases.

To deal with the above issues Gradient Descent is typically used in the form of Stochastic Gradient descent (SGD), an other variance of the algorithm. SGD computes the the gradient using only a random selected sample of the dataset and not the whole data. Hence parameters are updated at every step based on this relatively small sample. This method performs frequent updates that are usually much faster than the original and demand much less memory resources. Furthermore the methodology produce many random fluctuation on the loss function and the computed gradients. Those fluctuations allow the algorithm to escape from a local minimal and allow it to jump to a new and potentially better spot of the training space. Additionally the promote a better generalisation on external Data.

$$w \leftarrow w + \Delta w \quad \Delta w = -\gamma \frac{\partial L(\mathcal{D}_b|W)}{\partial w} \quad where\ \mathcal{D}_b \subset \mathcal{D} \tag{1.9}$$

1.2.0.1 SGD variants

Beyond the classical gradient descent in the same family there is a number of more advanced algorithms which are based on the same idea. An example is the the Gradient Descent with momentum. Gradient descent with momentum depends on two training parameters the learning rate γ just like the simple gradient descent and the new momentum parameter m. The introduction of the momentum makes the algorithm updates the weights by using not only the current gradient, but also the recent trends of the loss function. The exact effect of the momentum parameter is defining the dependence of the updates on the previous values. The range of the momentum parameter is $0 < m < 1$, 0 momentum corresponds to no dependence, while 1 corresponds to 100% dependence on the previous values and therefore no update of the gradient. Specifically the value of the Δw_i at epoch t is calculated via the equation 1.10.

$$\Delta w^t = m\Delta w^{t-1} + (1-m)(-\gamma \frac{\partial L}{\partial w^t}) \tag{1.10}$$

The usage of the momentum can beneficially affect the training procedure in various ways. Firstly it can accelerated the convergence especially in areas of the loss surface with lower gradient values. Further because of this feature it can also help the algorithm to escape from local minimals that would trap the algorithm and stop the training if ordinary gradient descent was used. Additionally momentum makes the training more stable and reduces the instillations of the loss values over the time.

Some other variants of the GD use even more parameters such as additional moments , decay rates normalization parameters and many more. Characteristic examples of these sophisticated and complex training algorithms are :

- Adaptive Gradient Algorithm(Adagrad) (Ruder, 2016; Mukkamala &Hein, 2017)
- Root Mean Square Propagation(RMSprop)(Mukkamala &Hein, 2017)
- Adadelta (Zeiler, 2012)
- Adaptive Moment Estimation (Adam)(Kingma &Ba, 2014a)

Adagrad algorithm adapts the learning rate to the parameters and performs larger updates for infrequent and smaller updates for the most frequent parameters (Papamakarios, 2014). Adadelta, RMSprop and Adam works in a similar way and they additionally keep an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients. The latest with the correct selection of parameters seems to be a very reliable method for training.

1.2.0.2 Over-fitting

Let C_{gen} the general class for all possible input/output data and $\mathcal{D}$ the set of the given data, hence $\mathcal{D} \subset \mathcal{C}_{gen}$. As mentioned above the goal of the training procedure and Neural Networks in general is not to fit 100% the given set of data $\mathcal{D}$ but to build a model that accurately every possible subset $\mathcal{D}' \subset \mathcal{C}_{gen}$ of the given task . The situation on which the model improves its performance on $\mathcal{D}$ but fails to give good results on out of sample entries is called over-fitting .

In order to monitor model's generalization we typically split the data $\mathcal{D}$ into 3 different sub-sets $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{valid}, \mathcal{D}_{test}\}$. Each one of those includes a set of inputs and the corresponding outputs.The main dataset $\mathcal{D}_{train}$ is called Training Data and includes the data on which the gradient descent will run and train the model. We use $\mathcal{D}_{valid}$ (validation set) for evaluating model's performance in a regular basis. Convergence on the validation set is the main criterion to conclude the training procedure. After finding the optimal parameters the final evaluation is done using the test set $\mathcal{D}_{test}$.

Over-fitting is a major problem for deep learning and various other machine learning algorithms. Fortunately, there are many methodologies that can help with avoiding or reducing over-fitting. A very popular and effective technique is the use of dropout layers (Srivastava et al., 2014). The functionality of Dropout is to drop away some units of the neural network during the training phase. More specifically it temporarily removes neurons form the network, along with all its incoming and outgoing connections. That is performed by using a random masking vector $\boldsymbol{m} \in \mathbb{Z}_2^n$ sampled independently from a Bernoulli Distribution that change the formulation of a single layer to.

$$\boldsymbol{h} = \boldsymbol{m} \circ \sigma(\boldsymbol{W}\boldsymbol{x} + b) \ , \quad \boldsymbol{m}_i \sim Bernoulli(p) \tag{1.11}$$

where $\circ$ represents element-wise multiplication.

This practices limit the gradient flow only to a number of neurons and allows the model to generalize more efficiently. The choice of which units to drop is purely random and the probability p to for each unit is given as a hyper-parameter.

Another equally popular and effective technique is weight regularization. This methods suggest adding an auxiliary extension on the Loss function that penalties high values of network's weights. Typically the additionally loss refers to mean squared or mean absolute values of the weights or a linear combination of those.

$$L_{model}(\mathcal{D}, \boldsymbol{w}) = L_{pred}(\mathcal{D}|\boldsymbol{w}) + L_{reg}(\boldsymbol{w}) \tag{1.12}$$

$$L_{reg}(\boldsymbol{w}) = \alpha \sum_i w_i^2 + \beta \sum_i |w_i| \tag{1.13}$$

Besides dropout layers and weight regularization there many other parameters that improve the over-fitting issue. An obvious example is an increased dataset size or the artificial augmentation of data. Finally we have to note that sophisticated stochastic methodologies such as variational Bayesian approaches provide better generelization and therefor are less prone to over-fitting.

1.3 Activation Functions

If a NN use only linear multiplication between the layers then regardless the size and the complexity of the architecture the network will be a simple linear model able to approximates only a very brief spectrum of functions. The objective of the activation functions is to make the connections between neurons non-linear extend the capability of the model. There are numerous functions that can be used efficiently as activation functions. The correct selection of those depends on the architecture of the model (size of the layers,depth etc), as well as the task which the network is trained for. In common practice, an activation function should be monotonic, and differentiable for $x \in R$. Two of the most popular choices are sigmoid and tanh funtions,

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{1.14}$$

Sigmoid is a bounded function that get values in the range of $(0, 1)$, more specifically $\lim_{x\to+\infty} \sigma(x) = 1$ and $\lim_{x\to-\infty} \sigma(x) = 0$. Beside the standard usage as an activation function sigmoid's mathematical properties make it a popular choice for gating and masking operations. Additionally is the standard output function for binary classification tasks where the output is probability $p \in [0, 1]$. Similar to sigmoid tanh is also a bounded funtion but in contrast to the previous it is zero centered ($tanh(0) = 0$) and $\lim_{x\to+\infty} tanh(x) = 1$, $\lim_{x\to-\infty} tanh(x) = -1$. The 2 functions have similar representative power and are especially popular in various cases.

The main disadvantage of the sigmoid funtion, and by extension of tanh, is the so called vanishing gradients problem. The gradient of the sigmoid is $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$ hence

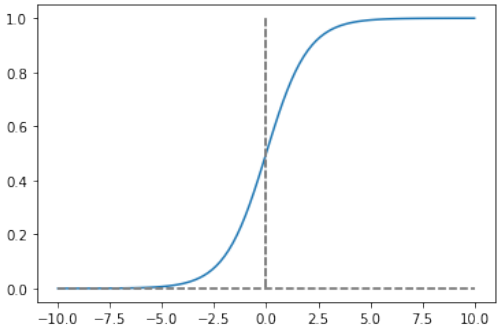

Figure 1.2: Sigmoid

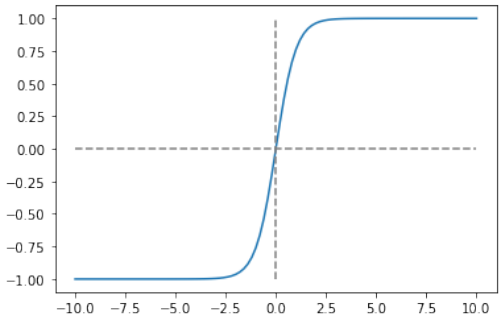

Figure 1.3: Tanh

for $|x| >> 0$ gradient appraches zero $\frac{d\sigma(x)}{dx} \sim 0$ and weight values are practically never updated. Furthermore the maximum value of the gradient is 0.25 which means that the gradients of the trainable parameters previous of the note are in best case 4 times smaller than the original (linear). Those effects get much bigger for deep architectures. Let a $f(x) = \sigma(\sigma(x))$ a trivial example of a deep model. The derivative is now $\frac{df(x)}{dx} = \frac{d\sigma(x)}{dx}(\frac{d\sigma(x)}{dx})$ tending to 0 much faster than before and in best case 16 times smaller than a linear model.

Rectified Linear Unit (ReLU) is currently the most frequently used nonlinearity in Deep Learning. Assuming an input $x \in R$ ReLU returns 0 if $x < 0$ and acts like an identity function in the opposite case .

$$ReLU(x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases} \tag{1.15}$$

ReLU is an especially simple activation function, technically nothing more than the concatenation of two linear functions. As a result is computationally significantly more efficient than sigmoid-like functions and it is considered suitable for Deep models avoiding the vanishing gradients (Krizhevsky et al., 2012). On the other hand the activation can blow up reaching extremely large numbers with no constrain while on the other extreme if activation get below zero the connection is technically dead with zero gradient.

Literature is full of Relu variants with slightly more sophisticated format. Such examples are leaky relu and elu.

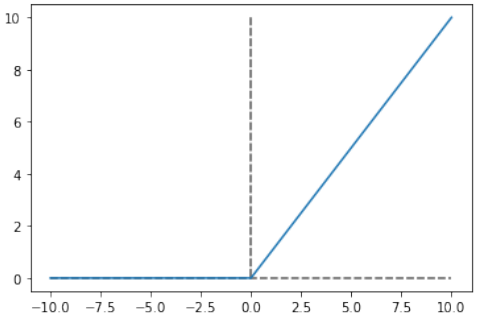

Figure 1.4: ReLU

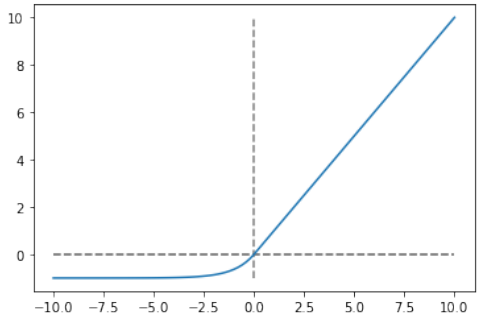

Figure 1.5: ELU

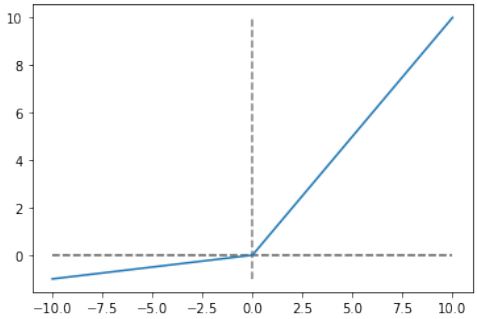

Figure 1.6: leakyReLU

$$leakyReLU(x) = \begin{cases} 0.1x & x < 0 \\ x & \text{otherwise} \end{cases}, \quad ELU(x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & \text{otherwise} \end{cases} \tag{1.16}$$

Both are designed to fix obstacles like the dying ReLU problem by alternate the zero value for $x < 0$. These formulations while yield close to zero values provide non-zero gradients allowing models to return from dead situations.

Chapter 2

Bayesian Deep Learning

Bayesian deep learning is an approach that integrates Bayesian inference with artificial neural networks, providing to these networks several beneficial properties. This fusion results in Bayesian Neural Networks (BNNs) that have the intrinsic ability to incorporate uncertainty directly into the deep learning models. In contrast to traditional neural networks with fixed parameters, BNNs consider parameters as probability distributions. This approach inherently embeds uncertainty within the model, enhancing its robustness while maintaining performance. The probabilistic nature of these networks offers a more nuanced and flexible framework for learning, making them particularly suitable for a wide range of applications.

Additionally, BNNs exhibit a significant resistance to overfitting, a notable challenge in deep learning. Due to their stochastic characteristics, stemming from probabilistic parameterization, these networks are less likely to overfit to the training data. This attribute is particularly advantageous in complex or over-parameterized models. The advantages of Bayesian Neural Networks extend beyond just avoiding overfitting; they also include enhanced accuracy, uncertainty awareness, improved generalization capabilities to new data, and more efficient memory usage. This chapter aims to provide a comprehensive examination of these features, focusing on the theoretical foundations of Bayesian deep learning. It seeks to clarify how the unique properties of BNNs can be effectively leveraged to optimize both the performance and efficiency of neural network models.

2.1 Bayesian Methods

Lets consider $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ to be a parametric model defined by a set of parameters $\boldsymbol{\theta}$. Unlike standard (Frequentist) statistical modeling, where these parameters are assumed to be fixed constants, Bayesian inference treats $\boldsymbol{\theta}$ as random variables. This perspective does not imply genuine randomness, but rather encompasses the uncertainty about the true values of the parameters.

The cornerstone of this methodology is Bayes' theorem, reads as:

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{D})} \tag{2.1}$$

where $\boldsymbol{\theta}$ represents the parameters under investigation and $\mathcal{D}$ signifies the observed data.

In Bayesian inference, the posterior distribution $P(\boldsymbol{\theta}|\mathcal{D})$ of the parameters is derived by employing the conditional probability $P(\mathcal{D}|\boldsymbol{\theta})$ and the prior probability $P(\boldsymbol{\theta})$. The prior probability embodies our initial understanding before any data observation. Conversely, $P(\mathcal{D}|\boldsymbol{\theta})$ integrates the data into the model. Consequently, inferences are drawn by merging this prior knowledge with the observed data $\mathcal{D}$. The denominator, $P(\mathcal{D})$, signifies the marginal likelihood, serving essentially as a normalizing constant.

The computation of $P(\mathcal{D})$ involves an integral that often poses a significant algebraic challenge and leads to computational inefficiency:

$$P(\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.2}$$

The integration of this marginal likelihood is crucial for deriving an analytical form of the posterior probability, but this is rarely feasible. Hence, in such scenarios, we resort to various sampling and approximation algorithms.

2.1.1 MCMC Sampling

Markov Chain Monte Carlo (MCMC) represents a collection of advanced stochastic sampling algorithms designed for accurately approximating a broad spectrum of distributions,

symbolized as $P(\boldsymbol{x})$, with $\boldsymbol{x} \in R^m$. These algorithms are fundamentally based on constructing Markov chains that are specifically engineered to achieve an equilibrium distribution, denoted as $P_e(\boldsymbol{x})$. This equilibrium distribution is tailored to approximate closely the desired target distribution $P(\boldsymbol{x})$. By employing such Markov chains, one can effectively sample from $P(\boldsymbol{x})$ by methodically accumulating the states of the chain. This subsection aims to provide a detailed exploration of some prominent MCMC algorithms, highlighting their unique characteristics and applications.

2.1.1.1 Metropolis–Hastings

The Metropolis-Hastings algorithm is a highly esteemed member of the MCMC family, initially developed to tackle complex statistical physics problems (Metropolis et al., 1953). This algorithm is particularly popular for its adaptability and straightforward implementation. Its core capability is to draw samples from a specified distribution $P(\boldsymbol{x})$ using a function $p(x)$, which is proportional to $P(\boldsymbol{x})$, $p(\boldsymbol{x}) \sim P(\boldsymbol{x})$. This feature is crucial in the field of Bayesian inference, where normalizing factors often pose significant computational challenges, sometimes rendering them infeasible.

Initiating the Metropolis-Hastings algorithm involves selecting an arbitrary starting point $\boldsymbol{x}_0$ within the R^m dimensional space. Subsequently, an integral part of the process is choosing a suitable candidate-generating distribution $Q(\boldsymbol{x}|\boldsymbol{x}_t)$. This distribution is employed to generate potential future states. The chosen distribution $Q(\boldsymbol{x}|\boldsymbol{x}_t)$ must depend solely on the current state and exhibit symmetry, adhering to the condition $Q(\boldsymbol{x}'|\boldsymbol{x}) = Q(\boldsymbol{x}|\boldsymbol{x}')$. While various distributions meeting these criteria can be used, a prevalent and practical choice is a Gaussian distribution, where the mean (μ) is set to the current state $\boldsymbol{x}_t$, thus transforming the process into a form of Gaussian random walk.

Once $p(\boldsymbol{x})$, $Q(\boldsymbol{x}|\boldsymbol{x}_t)$, and the initial state $\boldsymbol{x}_0$ are established, the algorithm proceeds to define a loop for the sampling process. Each iteration begins by generating a candidate state $\boldsymbol{x}'$, following the distribution $Q(\boldsymbol{x}|\boldsymbol{x}_t)$. The next step involves calculating an acceptance ratio, $a = \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x}_t)}$, to determine the viability of accepting this new candidate state. This ratio a is then compared with a randomly generated number z from a uniform distribution

$U[0,1]$. If $a > z$, then the candidate state $\boldsymbol{x}'$ is accepted as the new state ($\boldsymbol{x}_{t+1} = \boldsymbol{x}'$). If not, the algorithm repeats the process from the initial step.

Despite the proven effectiveness of MCMC algorithms in closely approximating the target distribution $P(\boldsymbol{x})$, they are accompanied by significant drawbacks, mainly related to processing speed and computational resource demands. A primary challenge is the duration required for the algorithm to achieve an equilibrium distribution that closely matches the target $P(\boldsymbol{x})$. This convergence can be especially protracted when the initial point $\boldsymbol{x}_0$ is not well-chosen, necessitating numerous iterations to reach a state of convergence. Additionally, due to the inherent characteristics of the random walk strategy employed, a considerable number of samples is often required. This high sample count is a direct consequence of the significant correlation typically observed between consecutive states in the Markov chain, presenting further challenges in the efficient implementation of MCMC algorithms.

2.1.2 Variational Inference

Variational Inference (VI), as presented by Wainwright &Jordan (2008), introduces a distinctive strategy that diverges from the traditional Markov Chain Monte Carlo (MCMC) methods. VI employs an alternative tactic by using an auxiliary distribution, denoted as $q(\boldsymbol{\theta})$, to approximate the true but computationally challenging posterior distribution $P(\boldsymbol{\theta}|\mathcal{D})$.

At its core, VI involves selecting a specific family of distributions, labeled Q, which are characterized by their well-defined analytical forms. The aim here is to identify the particular member $q \in Q$ that offers the closest approximation to the actual posterior distribution P. This selected family Q is typically parameterized by a set of parameters $\boldsymbol{\phi}$. Thus, the task within VI transforms into an optimization challenge, where the objective is to find the optimal parameter set $\boldsymbol{\phi}$ that results in the most accurate approximation of the posterior distribution.

This optimization is achieved by minimizing the divergence between the true posterior $P(\boldsymbol{\theta}|D)$ and the variational approximation $Q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. Common practice in VI involves

expressing this divergence using the Kullback–Leibler (KL) divergence, which is a measure of how one probability distribution diverges from a second, reference probability distribution. The KL divergence in this context is defined as:

$$KL(P(\boldsymbol{\theta}|\mathcal{D})||Q_{\boldsymbol{\phi}}(\boldsymbol{\theta})) = E[P(x)\log(\frac{P(\boldsymbol{\theta}|D)}{Q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}) \tag{2.3}$$

This equation quantitatively measures the difference between the proposed variational distribution and the actual posterior.

2.1.2.1 Kullback–Leibler Divergence

To further understand the role of Kullback–Leibler divergence (KL) in Variational Inference, let's delve into its properties and usage as a tool for comparing probability distributions. The KL divergence is defined differently for discrete and continuous probability spaces, as shown in the following equations:

For discrete probability spaces, the KL divergence is defined as:

$$\mathrm{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{2.4}$$

For continuous probability spaces, it is defined as:

$$\mathrm{KL}(P \parallel Q) = \int_{-\infty}^{\infty} P(x) \log \left(\frac{P(x)}{Q(x)} \right) dx \tag{2.5}$$

The KL divergence's values range from 0 to infinity, with higher values indicating greater divergence between the distributions. If P and Q are identical for all x ($P(x) = Q(x); \forall x$), then $KL(P||Q) = 0$. Conversely, if P and Q are completely dissimilar, the KL divergence tends towards infinity.

Computing the KL divergence for continuous distributions, especially over unbounded multi-dimensional domains, can be computationally intensive or even infeasible. However, for certain families of distributions, such as Gaussian distributions, the KL divergence can be analytically derived, avoiding costly integration. For example, the KL divergence

between two Gaussian distributions $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$ can be expressed as a function of their parameters:

$$\mathrm{KL}(N(x|\mu_1, \sigma_1) \parallel N(x|\mu_2, \sigma_2)) = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1\right] \tag{2.6}$$

In cases where an analytical solution for the KL divergence is not feasible, Monte Carlo estimation offers a practical alternative. This method is particularly useful when we can efficiently sample from the distribution $P(x)$. The Monte Carlo estimation of the KL divergence involves approximating the integral by averaging the log-ratio of the probabilities of sampled points. The approximation formula can be expressed as follows:

$$\mathrm{KL}(P \parallel Q) \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{P(x_i)}{Q(x_i)}\right), \quad x_i \sim P(x) \tag{2.7}$$

Here, N represents the number of samples drawn from the distribution $P(x)$, and x_i are the individual samples. This Monte Carlo approach calculates an approximate value of the KL divergence by averaging the log-ratio of probabilities for these samples. Although this method may not provide the exact value of the KL divergence, it offers a computationally feasible way to estimate it, especially in high-dimensional spaces or when dealing with complex distributions.

In conclusion, Variational Inference leverages the KL divergence as a metric to optimize the approximation of the true posterior distribution. The use of KL divergence, either through analytical calculation for certain distribution families or via Monte Carlo estimation for more complex cases, is central to the effectiveness of VI in probabilistic inference, especially in scenarios where direct computation of the posterior distribution is impractical.

2.1.2.2 Evidence Lower Bound

To optimize $KL(P||Q)$, access to the intractable $P(\boldsymbol{\theta}|\mathcal{D})$ and $P(\mathcal{D})$ is required, leaving the problem unsolved. A solution to this problem lies in maximizing the Evidence Lower Bound (ELBO), which is equivalent to minimizing the KL divergence between q and p.

The ELBO, denoted as $L(\phi)$, is given by:

$$L(\boldsymbol{\phi}) = \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log p(\mathcal{D}|\boldsymbol{\theta}) d\boldsymbol{\theta} - KL\left(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \parallel P(\boldsymbol{\theta})\right) \tag{2.8}$$

The former expression represent the lower bound of the log-evidence $log(P(\mathcal{D}))$ hence $log(P(\mathcal{D})) \geq L(\boldsymbol{\phi})$. The latest inequality can be easily been proved:

$$\begin{aligned}
&KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) = +\int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{P(\boldsymbol{\theta}|\mathcal{D})} d\boldsymbol{\theta} = -\int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log \frac{P(\boldsymbol{\theta}|\mathcal{D})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} d\boldsymbol{\theta} \Rightarrow \\
&KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) = -\int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log \frac{P(\boldsymbol{\theta})P(\mathcal{D}|\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})P(\mathcal{D})} d\boldsymbol{\theta} \Rightarrow \\
&KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) = -\int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log \frac{P(\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} d\boldsymbol{\theta} + \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log(P(\mathcal{D})) d\boldsymbol{\theta} - \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log(P(\mathcal{D}|\boldsymbol{\theta}) d\theta \Rightarrow \\
&KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) = +KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) - log(P(\mathcal{D})) - \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log(P(\mathcal{D}|\boldsymbol{\theta})) d\boldsymbol{\theta}
\end{aligned} \tag{2.9}$$

From the 2.9 and given that $KL(Q||P) \geq 0$ we get :

$$\begin{aligned}
log(P(\mathcal{D})) - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta}|\mathcal{D})) = \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) log(P(\mathcal{D}|\boldsymbol{\theta})) d\boldsymbol{\theta} - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||P(\boldsymbol{\theta})) \Rightarrow \\
log(P(\mathcal{D})) \geq L(\boldsymbol{\phi})
\end{aligned} \tag{2.10}$$

Furthermore since $log(P(\mathcal{D}))$ is a constant 2.9 shows the initial statement that maximizing ELBO minimizes $KL(q_{\boldsymbol{\phi}}||P(\boldsymbol{\theta}||\mathcal{D}))$.

2.2 Bayesian Neural Networks

Conventional Neural networks learn by finding the values of their trainable parameters $\boldsymbol{\theta} = \boldsymbol{w}$ for which they achieve the optimal performance for a specific task T. In that frequentist case, the final product is a set of point estimations for all these parameters. Bayesian deep learning is an alternative approach of NNs where the trainable parameters are approached not as fixed values but as random variables in a Bayesian fashion (Jospin et al., 2021; Blundell et al., 2015).

In this context, we introduce distributions over all network parameters $p(\boldsymbol{w})$. We now target not to find an optimal value for θ but to properly adjust the weight distributions on the given observations. Throughout the training phase we conclude to a data-aware posterior distribution $p(\boldsymbol{w}|\mathcal{D})$ for all the model's parameter's. By the usage of Bayesian rule the posterior is expressed as (Jospin et al., 2021; Heckerman, 2008; Kingma &Welling, 2013) :

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} \tag{2.11}$$

The complexity of Deep Models and the enormous number of the parameters make posterior inference and prediction on unseen data are intractable. As a result we need to some sort of approximation technique to both train and estimate such models. Typically these techniques include some sort of Monte Carlo treatment. Therefore running a BNN includes a stochastic sampling from the Postrerior $P(\boldsymbol{w}|D)$ (or an its approximation) that make BNN itself a stochastic procedure representing sampling from a distribution $P_{BNN}(y|x, p(\boldsymbol{w}))$

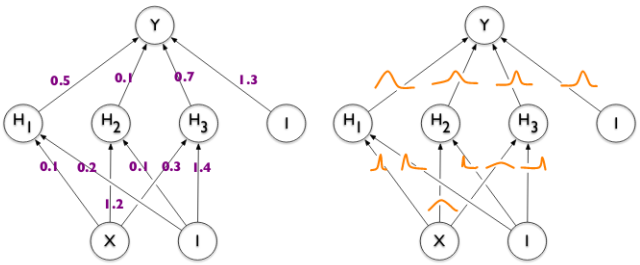

Figure 2.1: Left: Conventional Neural network, Right: Bayesian Neural Network

While MCMC algorithms have been proposed and used for BNN from various researches their computational cost make them an inferior method. Variatioanal inference on the other hand has been proved an exceptional choice.

2.2.1 Variatonal Bayes NN

To deal with posterior inference over the complex structures of BNNs we need to employ approximate techniques. The variational approach has been proved the most efficient

and effective way to work. As we discussed before we need to select a family of posterior distributions that would describe well the true and proper corresponding priors. In the Bayesian Deep Learning concept we then fit the model to minimize the prior-posterior deviation.

In the VI schema we employ an variational posterior distribution to approximate the true but intractable posterior; let q such a distribitution parameterized by variational parameters $\boldsymbol{\phi}$. We now want maximimize and Evidence Lower Bound that constitutes the lower bound. ELBO is usually constructed from the summation of CrossEntropy and the Kullback–Leibler divergence between priors and variational posteriors. The first term is model's primary metric while the second one is a regularization term.

ELBO is constructed from the summation of two parts. The fist term is the expected log-likelihood, maximizing this part is equilecant to searching for a variational distribution that explains the data well. The second term is the prior-posterior KL divergenvr which acts as regularization keeping variotional posterior close to the pior.

$$L(\phi) = \int_{\boldsymbol{w}} q_{\boldsymbol{\phi}}(\boldsymbol{w}) log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{w})}{p(\boldsymbol{w})} d\boldsymbol{w} + KL(q_{\boldsymbol{\phi}}(w)||p(\boldsymbol{w})) \tag{2.12}$$

Calculation ELBO analytically is impossible since the expected likelihood is intractable. However it can be well approximated via a Monte Carlo approach. The necessity and the success of the Monte Carlo approach makes the term Stochastic Neural Networks interwoven with these Bayesian methods.

In order to apply the backpropagation algorithm effectively in a Monte Carlo setting, it is necessary to efficiently compute the gradients resulting from differentiating the ELBO with regards to the networks parameters. Recent literature in the field is in large part concerned with different ways of solving this problem; Bayes by Backprop with reparameterizations tricks being the most famous methods.

2.2.2 Reparameterization trick

In order to train a Variational BNN we need to draw samples from the posteriors in way that concurrently allows the gradient flow to pass and update the parameters. Reparameterization trick (Kingma &Welling, 2013) is a way to rewrite this expectation so that the distribution with respect to which we take the gradient is independent of trainable parameters $\boldsymbol{\phi}$, and that by making the stochastic element in $q_{\boldsymbol{\phi}}$ independent of $\boldsymbol{\phi}$.Technically we employ an auxiliary and purely random variable z and a deterministic function $t(\boldsymbol{\phi}, z)$ for which:

$$w = t(z, \boldsymbol{\phi}) \;\; z \sim p(z) \text{ is equilevant to } w \sim q_{\boldsymbol{\phi}}(w) \tag{2.13}$$

and therefore:

$$\int_w f(w) q_{\boldsymbol{\phi}}(w) dw = \int_z f(t(\boldsymbol{\phi}, z)) p(z) dz \tag{2.14}$$

As a characteristic example we now present the case of Gaussian Distribution.

2.2.2.1 Gaussian Distribution

Gaussian distributions are notably prevalent in modeling the posteriors of weights in various probabilistic models. Their use is partly attributed to the tractability of their Kullback–Leibler (KL) divergence and the efficiency of the reparametrization trick associated with them. Consider a variational posterior of the weights represented by a Gaussian distribution $N(w|\mu, \sigma)$. Utilizing the basic algebraic properties of Gaussian distributions, we can express $N(w|\mu, \sigma)$ as $\mu + \sigma N(w|0, 1)$. This expression simplifies the process of sampling from $N(w|\mu, \sigma)$, which is effectively equivalent to drawing a sample z from $N(w|0, 1)$, scaling it by σ, and then adding μ.

$$w = \mu + \sigma z \;\; where \;\; z \sim N(0, 1) \tag{2.15}$$

In that way, we separate the random generator from the trainable parameters and enable training through gradient descent.

2.2.2.2 Discrete Distributions

In several Stochastic DL setups(Panousis et al., 2019b), we often need to draw samples from trainable discrete distributions. While not identical the case shares many properties of the other reparametrization tricks. An elegant way to do that is the Gumbel Softmax relaxation trick (Jang et al., 2016).

$$\begin{aligned} \boldsymbol{\xi} &= \frac{\exp\left((\log \boldsymbol{\eta} + \boldsymbol{g})/T\right)}{\sum_{i=1}^{U} exp((\log \eta_i + g_i)/T)} \\ \boldsymbol{g} &= -log(-\log \boldsymbol{z}),\ \boldsymbol{z} \sim \mathrm{U}(\mathbf{0}, \mathbf{1}) \end{aligned} \tag{2.16}$$

This technique is designed to offer low-variance gradients and smooth ELBO convergence during the training phase but almost perfectly discrete(one-hot) samples during inference. The hardness of GS is controlled by the temperature hyper-parameter T; a high T value implies smooth gradient while low temperatures yield the true discrete posterior.

2.2.3 Training and Inference

2.2.3.1 Bayes by Backprop

Lets now examine the exact training algorithm we use for BNNs(Back &Keith, 2019; Heckerman, 2008). For the rest part we assume the usage of Gaussian $N(\boldsymbol{\mu}, \boldsymbol{\sigma})$ as the variational posterior and a spherical Gaussian $p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ as a proposed prior distribution; that is by far the most frequently used case.

The loss function we look to minimize to train such networks is the negative ELBO.

$$Loss(\boldsymbol{w}, \phi) = \int_{w} q_{\boldsymbol{\phi}}(\boldsymbol{w}) p(\mathcal{D}|\boldsymbol{w}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{w})||p(\boldsymbol{w})) \tag{2.17}$$

The integral of the first part is unapproachable and to deal with it we need to emlpoy an MC approximiation and rewrite the Loss function as:

$$\int_{\boldsymbol{w}} q_{\boldsymbol{\phi}}(\boldsymbol{w}) p(\mathcal{D}|\boldsymbol{w}) \approx \frac{1}{L}\sum_{l=1}^{L} p(\mathcal{D}|\boldsymbol{w}^l) \;\; where \;\; \boldsymbol{w}^l \sim q_{\boldsymbol{\phi}}(\boldsymbol{w}). \tag{2.18}$$

Similar approximation can be applied to KL term as well. Although given our assumption for Gaussian posterior calculations can become ever simpler. We can reduce the prior-posterior kl divergence to a close form. Based on the previous discussion and 2.6:

$$\mathrm{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{w})||p(\boldsymbol{w})] = \mathrm{KL}(\phi = \{\mu, \sigma\}) = [\frac{\boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2}{2} - \log \boldsymbol{\sigma} - \frac{1}{2}] \tag{2.19}$$

Now we want to optimize the Loss function with respect to $\boldsymbol{\phi} = \{\mu, \sigma\}$ and that through gradient descent and backpropagation. Consequently we need to calculate the Gradient of Loss with respect to the parameters. However the stochasticity of the monter carlo appraximation blocks the access to the internal parts of a network 2.18. Thanks to existing of the reparametrizitiotion trick we can set $w = t(z, \boldsymbol{\phi})$ and get:

$$\int_{\boldsymbol{w}} q_{\boldsymbol{\phi}} v(\boldsymbol{w}) p(\mathcal{D}|\boldsymbol{w}) d\boldsymbol{w} = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p(\mathcal{D}|t(\boldsymbol{z}, \phi)) d\boldsymbol{z} \approx \frac{1}{L}\sum_{l=1}^{L} p(\mathcal{D}|t(\boldsymbol{z}^l, \phi = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}))$$
$$where \;\; \boldsymbol{z}^l \sim N(0, \boldsymbol{I}). \tag{2.20}$$

By utilizing the equations 2.19 and 2.20 we are able to approximate well the Loss with an expression depended exclusively on the parameters $\boldsymbol{\phi}$ and the auxiliary z. Now it is possible to effectively calculate the gradient with respect to any variational parameter $\frac{\partial Loss}{\partial \boldsymbol{\phi}}$.

$$\begin{aligned}
\frac{\partial Loss}{\partial \boldsymbol{\phi}} &= \frac{\partial}{\partial \boldsymbol{\phi}} \int_z p(z) p(\mathcal{D}|t(z, \boldsymbol{\phi})) dz + \frac{\partial KL(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \Rightarrow \\
\frac{\partial Loss}{\partial \boldsymbol{\phi}} &= \frac{\partial}{\partial \boldsymbol{\phi}} \frac{1}{L} \sum_{l=1}^{L} p(\mathcal{D}|t(z^l, \boldsymbol{\phi})) + \frac{\partial KL(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \Rightarrow \\
\frac{\partial Loss}{\partial \boldsymbol{\phi}} &= \frac{1}{L} \sum_{l=1}^{L} \frac{\partial p(\mathcal{D}|t(z^l, \boldsymbol{\phi}))}{\partial \boldsymbol{\phi}} + \frac{\partial KL(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \;, \quad z^l \sim p(z)
\end{aligned} \tag{2.21}$$

We have to note that theoretically in order to achieve a strong Monte Carlo approximation the sample size L needs to be relatively high. However in practice it has been proved that even for $L = 1$ we are able to get low variant gradients and train a BNN effectively. This is partly due the mechanism of the Stochastic Gradient Descent that gets gradient not on the whole dataset $\mathcal{D}$ but on N different subsets of $\mathcal{D}$ with size $M = \frac{D}{N}$. Thankfully this practice benefits the MC version of the expected log-likelihood providing with a sufficient sample.

2.2.3.2 Inference

In the context of Bayesian Neural Networks (BNNs), the inference algorithm adopts a more streamlined approach. The primary task involves generating samples from the variational posteriors, symbolized as $w \sim q_\theta(w)$, and subsequently deploying these samples within the network. To elaborate, the inference process entails repeatedly sampling from $q_\theta(w)$, specifically S times. The next step involves calculating the average of the output logits for each sample, a method known as Bayesian averaging. The value of S plays a pivotal role: a higher S generally leads to improved accuracy but at the expense of a longer inference time. In contrast, a smaller S can increase the variance of the output while reducing the computation time in a linear fashion. Selecting the optimal S value largely depends on the specific characteristics of the model and the nature of the task being addressed.

This concept can be mathematically represented as:

$$F_{q_{\boldsymbol{\phi}}}(x) = \frac{1}{S}\sum_{i=0}^{S} F_{\boldsymbol{\theta_i}}(x) \ , \ \theta_i \sim q_{\boldsymbol{\phi}}(w) \tag{2.22}$$

Here, $F_{q_{\boldsymbol{\phi}}}(x)$ denotes the final inferred output, which is the average of the outputs $F_{\theta_i}(x)$, each computed from a different sample θ_i drawn from the posterior $q_{\boldsymbol{\phi}}(\theta)$.

Chapter 3

Sequential Data and Transformers

Sequential data modeling is an important and broad area of machine learning, related to numerous real-world applications. The term "sequential data" refers to any form of input/output data that evolves over a time axis or has a dependency on a time-like parameter, whether discrete or continuous. Notable examples of sequential data include natural language and videos. Natural language relies on the sequential order of words to convey meaning, while videos consist of image sequences that capture motion over time. Both are crucial for deep learning, the foremost approach for analyzing such data.

Deep learning has presented many significant advancements in sequential data and NLP over the last years, yet the introduction of Transformer networks arguably represents the most noteworthy leap forward. Distinct from prior models, Transformers exclusively utilize attention mechanisms, notably self-attention, removing the older dependence on recurrent methods. This advancement facilitates a dynamic comprehension of the varying significance of different data segments, thereby enhancing performance and modeling efficiency, particularly for extended sequences, making them exceptionally suited for processing languages, videos, and many other data formats.

This chapter introduces the essential background on sequential data and natural language processing (NLP), establishing the necessary groundwork for the chapters that follow. It traces the historical development of relevant deep learning techniques, concluding to the introduction of Transformer networks. The chapter provides a detailed examination of Transformers, exploring the core mechanisms, applications, and advantages over preceding approaches. Additionally, we briefly discuss their latest variants and applications in NLP and beyond.

3.1 Recurrent Neural Networks

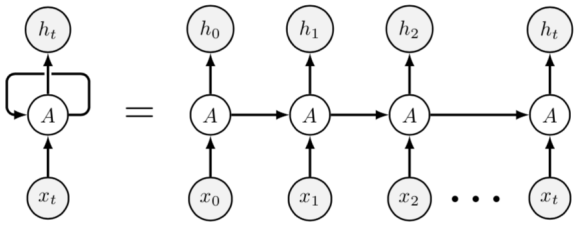

Figure 3.1: Recurrent Neural Networks

Recurrent Neural Networks (RNNs), as identified by Medsker &Jain (2001), are specifically designed architectures for processing sequential data. These networks distinguish themselves through connections that span across the temporal dimension, forming a directed graph. This configuration endows RNNs with a form of internal memory, enabling dynamic temporal behavior. This feature is particularly advantageous for applications in text, speech, and video analysis. In contrast to dense layers, RNNs process information not just from the present input, but also incorporate feedback previous network's step. Mathematically, the output, or hidden state h, of an RNN is given by:

$$\boldsymbol{h}_t^i = F(\boldsymbol{x}_t^{i-1}, \boldsymbol{h}_{t-1}^i | \ \theta) \tag{3.1}$$

The function F in RNNs takes various forms, defining different versions of the RNN architecture. The most basic form, often referred to as a vanilla RNN, operates by concatenating $\boldsymbol{x}^{i-1}t$ and $\boldsymbol{h}^i t-1$, and then processing them in a manner similar to dense layers. This approach is encapsulated in the following equation referred to as 3.2:

$$F(\boldsymbol{x}_t, \boldsymbol{h}_{t-1} | W, U, b) = \tanh\left(W\boldsymbol{x}_t + U\boldsymbol{h}_{t-1} + \boldsymbol{b}\right) \tag{3.2}$$

RNNs remain widely used for sequence-based applications due to their simplicity and effectiveness. However, they are not without significant drawbacks. The sequential nature of RNNs impedes full parallelization of computations, leading to exponentially increased training and inference times. Another major issue is the vanishing gradient problem and

particularly in larger networks. This occurs when the gradients of earlier layers diminish to almost zero during backpropagation, resulting in extremely slow training processes.

3.1.0.1 LSTM

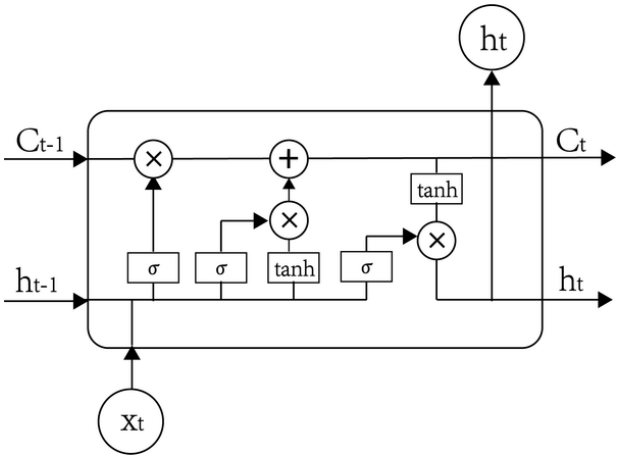

Figure 3.2: The LSTM Module.

Long Short Term Memory networks (LSTMs) are a kind of sophisticated RNNs and have the same chain like structure as the original. Although, unlike the simple recurrent unit which overwrites its content at each time-step, an LSTM unit is able to decide whether to keep the existing memory. Intuitively, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information(Chung et al., 2014). This functionality allows them to reduce the vanishing gradient problem.

The exact LSTM's formulation, while follows the 3.1 scheme, is significantly different than the original case. A regular RNN layer hidden unit h,has both the roles of the central output and the only memory-related representation. This approach confines the functionalities of the memory system especially for long sequences. The centre idea behind LSTM is the introduction of the cell state c, a vector purposed to act exclusively as the main memory channel of the network. The hidden unit still contributes in memory transferring but now in a limited and auxiliary way. Therefore we now have schematic equation 3.3

$$h_t = F(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1} | \ \theta \ ) \tag{3.3}$$

where θ the set of all the trainable parameters

LSTM features a complex internal memory processing system based on a series of gating operations. Throughout of those it can add, delete, and forget information from the cell state in a controlled way and leverage the stored memory to improve the predictions. Figure 3.2 makes a visual representation of the exact gating system. ysis of the whole unit we have a set of 3 mechanisms. The first one is based on a forget gate vector f which acts as a weight on old information and technically decides which parts to remember. Vector f at time step t is a function of the input x_t and the previous latent state h_{t-1}.

$$f_t = \sigma_g(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{3.4}$$

Subsequently there is a two phase mechanism that determines what new information are going to get stored on the cell state. More specifically it computes a vector of candidate values $\tilde{c}$ and an input gate vector i that decides which values of $\tilde{c}$ will be infused to the cell state. Similarly to f_t at time step t the i_t and $\tilde{c}$ vectors are:

$$\boldsymbol{i}_t = \sigma_g(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{3.5}$$

$$\tilde{\boldsymbol{c}}_t = \sigma_c(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{3.6}$$

The final cell state is computed as function of of the previous cell state and the new candidate values in the following way:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{c}_t \tag{3.7}$$

where $\circ$ element wise multiplication

The third mechanism aims to produce the final output of the model based on the input $\boldsymbol{x}$ and whole inherited memory. In order to do so it produce a candidate output vector $\boldsymbol{o}$ and combines it with the current cell state $\boldsymbol{c}_t$ in the following way.

$$o_t = \sigma_g(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{3.8}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \sigma_h(\boldsymbol{c}_t) \tag{3.9}$$

3.1.0.2 GRU

A third popular RNN vatiaion is the Gated recurrent unit (GRU) (Chung et al., 2014). GRU does not feature LSTM's cell state system (or equivalent). It follows the original, and simpler, one vector approach using only the previous hidden state. However GRU has a repeating unit significantly more advanced the vanilla case.

Two are gating vectors used, the reset gate vector $\boldsymbol{r}_t$ and the update gate vector $\boldsymbol{z}_t$ calculated through:

$$\boldsymbol{r}_t = \sigma_g(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{3.10}$$

$$\boldsymbol{z}_t = \sigma_g(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{3.11}$$

The first is used to determinate which part of the previous unit $\boldsymbol{h}_{t-1}$ should be included on a new candidate vector $\hat{\boldsymbol{h}}_t$. While the second one is used on the diffusion of $\hat{\boldsymbol{h}}$ into $\boldsymbol{h}_{t-1}$. The exact formulation for those is:

$$\hat{\boldsymbol{h}}_t = \tanh \boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h) \tag{3.12}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \circ \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \circ \hat{\boldsymbol{h}}_t \tag{3.13}$$

3.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of Artificial Intelligence and computational linguistics dedicated to the automated processing, understanding, and generation of human (natural) language. Natural language is usually expressed through speech or text. Speech-related NLP tasks include speech recognition, which converts audio signals into corresponding textual data, and speech synthesis, the reverse of this process.

Although speech is a form of natural language, the term NLP tends to focus more on textual data. Textual information is more easily understood by machines and provides a

more refined form of linguistic knowledge. Furthermore, the availability of data in text form is much larger and easier to store and handle.

Initially, the field relied exclusively or partly on rule-based methods, although these approaches have now been superseded by modern machine learning techniques. Deep Learning has emerged as the dominant methodology in NLP. The contemporary literature is filled with advanced neural network (NN) architectures adept at processing textual data. A common starting point for many of these models involves word tokenization and embedding.

Word Embedding

Tokenization is the process of dividing input and target sentences into basic units, often words or characters, with words being the preferred choice. Let V represent the comprehensive vocabulary of a dataset, comprising k distinct words and special characters. Each lexical unit is assigned a unique identifier $i \in [0, k-1]$. Representing these words in a neural network typically involves one-hot encoding, where each word $\boldsymbol{w}_i$ is represented as a k-dimensional vector $\boldsymbol{w} \in \{0,1\}^k$, with all elements set to zero except for the ith entry.

However, one-hot encoding has its limitations, as it does not reflect semantic relationships between words. The embedding phase aims to map these tokens into a more expressive vector space, usually through a trainable fully connected layer, possibly with a non-linear activation function:

$$\boldsymbol{w}_e = \text{Embedding}(\boldsymbol{w}_o|\boldsymbol{W}) = \sigma(\boldsymbol{w}_o^T\boldsymbol{W}), \tag{3.14}$$

$$\text{where } \boldsymbol{w}_o \in \text{one\_hot}^k \text{ and } \boldsymbol{w}_e \in \mathbb{R}^l. \tag{3.15}$$

This embedding space offers an effective way to quantitatively model a word's meaning. Words with similar meanings are projected in close proximity, while more complex semantic relationships can be expressed through specific geometric properties. The dimensionality l of the resulting vectors is a crucial tunable hyperparameter for a typical Deep NLP model. The embedding should be large enough to encapsulate the necessary information, as too small a size limits its expressive power, whereas an overly large space may lead to

high sparsity and fail to correctly model the required geometric properties, in addition to increasing computational costs.

In addition to trainable embeddings, some models utilize pre-trained embedding layers like GLOVE and Word2Vec, which are preferred in certain contexts.

Temporal Interaction

As previously elaborated, natural languages inherently adhere to sequential structures, requiring sentences as their basic organizational units for conveying meaning comprehensively. Consequently, analyzing isolated words falls short of extracting requisite information. Following the embedding phase, subsequent layers are invoked to capture the temporal dynamics within sentences. These components manage word-to-word interactions within sentences and form the foundational core of every Deep Learning model designed for NLP. In earlier successful forays, recurrent neural networks (RNNs) and their variants constituted the default choice for modeling natural language. RNNs process words sequentially within a sentence, albeit with specific formulations and architectures tailored to the task at hand.

The most common and simple NLP tasks encompass classification and regression, in which a given input sentence $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ necessitates prediction of either a class or a continuous real number. Sentiment analysis, a prominent application, involves contextual analysis of text data to predict positive or negative sentiments. Such tasks conventionally employ architectures featuring a linear embedding layer and a stack of one or more RNN layers. The final hidden vector of the ultimate RNN layer is subsequently conveyed to an additional linear/softmax layer for effecting predictions.

3.2.1 Sequence-to-Sequence Models

Sequence-to-sequence (*Seq2Seq*) models (Sutskever et al., 2014) constitute neural network architectures designed specifically for tasks wherein both input and output are sequential in nature. Prominent among these are natural language processing (NLP) applications like automatic dialogue systems (chatbots) and neural machine translation, esteemed as among the most prevalent and consequential within the NLP domain.

In essence, given a source sequence $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and a target sequence $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$, the principal objective of *Seq2Seq* models revolves around the extraction of pertinent features from $\boldsymbol{x}$ and the generation of the complete output estimation sequence $\hat{\boldsymbol{y}}$. The salient challenge here stems from the disparity in lengths between $\boldsymbol{x}$ and $\boldsymbol{y}$, precluding a direct one-to-one mapping between their constituents. Consequently, the model is compelled to construct the entirety of the output sentence predicated upon the entirety of the input sequence.

The foundational architecture of *Seq2Seq* models comprises two distinct constituents: the encoder and the decoder. The encoder, occupying the initial phase of these architectures, assumes responsibility for processing input sentences, extracting pivotal information, and generating latent representations $\boldsymbol{e} = f_e(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\boldsymbol{\theta}_e)$ that enclose all the needed information. Subsequently, the decoder, which receives these representations as input, endeavors to synthesize syntactically correct output sentences. Decoders conventionally operate in an auto-regressive mode, predicting sentences word by word, with each prediction step informed by its predecessors: $p(\boldsymbol{y}_i) = f_d(\boldsymbol{e}, \boldsymbol{y}_{1:i-1}|\boldsymbol{\theta}_d)$. Frequently, decoder training proceeds under the assumption of perfect recognition of previous steps, a practice known as teacher forcing.

In initial RNN-based models, the encoder's latent state was conveyed as a solitary vector to the decoder, serving as the RNN's memory unit. However, this approach was marred by multiple limitations. The intrinsic capacity of a single vector to encapsulate information is inherently restricted and frequently inadequate for a comprehensive depiction of the entire sentence. Particularly in the later phases of decoding, the initial input information could fade into obscurity. Additionally, RNNs encountered difficulties when handling protracted sequences. The advent of the attention mechanism emerged as an elegant solution

3.3 Attention Mechanisms

Attention mechanisms are techniques that enable neural networks to selectively focus on specific components of multidimensional input data. In contrast to Convolutional and Recurrent layers, attention dynamically selects relevant subcomponents for information

processing.

Given a query vector q and a set of key-value pairs, $k = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_m\}$ and $v = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$, attention works by mapping q, k, and v to an output vector $\boldsymbol{o}$. Typically, the output is a weighted sum of the values $\boldsymbol{v}$, where the weights are determined by a compatibility score between the query and the corresponding key:

$$\boldsymbol{o} = \sum_{i=1}^{m} w_i \boldsymbol{v}_i, \quad w_i = \text{score}(\boldsymbol{q}, \boldsymbol{k}_i) \tag{3.16}$$

Attention mechanisms can be categorized as either local or global. In the case of local attention, only a limited portion of the key and value vectors is considered during the scoring process. This approach allows for faster and more targeted computation by avoiding distant parts of the input. In contrast, global attention considers the entire sets of keys and values, allowing the model to explore and process information from all available sources.

For sequence-to-sequence models, attention has proven to be an exceptionally useful feature and is now an integral part of most state-of-the-art architectures. As mentioned earlier, traditional seq-to-seq models face challenges such as the vanishing gradient problem and difficulty in transmitting encoded representations to longer decoders. These issues arise due to the inherent limitations of RNN-only networks, where the only connection between the encoder and decoder is through the memory units passed from the first to the last. Attention has emerged as a solution to this problem, as it enables decoders to access the entire encoded sequence for making predictions.

Let $\hat{\boldsymbol{h}}$ and $\boldsymbol{h}$ represent the encoder's and decoder's hidden states, respectively. Following the general attention scheme, we model $\boldsymbol{h}$ as the query values ($\boldsymbol{q}$), and $\hat{\boldsymbol{h}}$ as both the keys ($\boldsymbol{k}$) and values ($\boldsymbol{v}$). This way, the output $\boldsymbol{o}$ is a weighted sum of all encoder hidden states, and the decoder selects which ones to 'attend' to at each step. The actual mechanism for calculating and utilizing $\boldsymbol{o}$ varies depending on the specific task and architecture. In the following sections, we will delve into two of the most influential attention-based seq-to-seq variations: Luong (Luong et al., 2015) and Bahdanau (Bahdanau et al., 2014).

3.3.0.1 Luong's Attention Mechanism

Luong's attention mechanism stands out as one of the most successful variations in the realm of attention mechanisms. Its popularity can be attributed to the remarkable quality of results it delivers and its adaptable design that seamlessly accommodates new architectural developments. What sets Luong's attention apart is its versatility, as it does not prescribe a specific scoring function but rather offers a range of choices.

Luong's attention mechanism presents three distinct scoring functions:

$$\text{score}(\boldsymbol{q}_t, \boldsymbol{k}_s) = \begin{cases} \boldsymbol{q}_t \cdot \boldsymbol{k}_s & \text{Dot} \\ \boldsymbol{q}_t \cdot \boldsymbol{W}_a \boldsymbol{k}_s & \text{General} \\ \boldsymbol{u} \cdot \tanh(\boldsymbol{W}_s[\boldsymbol{q}_t; \boldsymbol{k}_s]) & \text{Concat} \end{cases} \tag{3.17}$$

Any of these scoring functions can be employed to compute the alignment scores for the encoder's latest hidden states, which serve as key-value pairs. Following the standard formulation, the context vector c is calculated as a weighted sum of the encoder's hidden states, denoted as $\boldsymbol{h}_i$ with associated weights w_i:

$$c = \sum_i \boldsymbol{h}_i w_i \tag{3.18}$$

The query vectors are derived from the current hidden states of the decoder's latest layer, represented as $\boldsymbol{h}_t$. To combine the context vector with the hidden state, a straightforward concatenation layer is employed. Subsequently, the result is processed through a single hyperbolic tangent (tanh) activated layer, yielding the final representation vector $\boldsymbol{h}$. Consequently, the predicted distribution becomes:

$$p(\boldsymbol{y}_t | \boldsymbol{y}_{<t}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s \boldsymbol{h}) \tag{3.19}$$

The context vector is computed based on the alignment scores and follows the standard formulation.

3.3.0.2 Bahdanau's Attention Mechanism

Bahdanau's attention formulation was introduced in one of the pioneering papers that significantly advanced sequence-to-sequence translation through the incorporation of attention mechanisms. In this version, the scoring values are computed using a trainable feedforward layer, which takes joint query and key vectors as input. While this model adheres to the conventional practice of using the last hidden layer of the encoder for key and value vectors, it deviates from the norm by employing the previous hidden state h_{i-1} of the RNN as query values.

The scoring function is defined as follows:

$$score(\boldsymbol{q}, \boldsymbol{k}) = \text{softmax}(\boldsymbol{W}_s[\boldsymbol{q}; \boldsymbol{k}]) \tag{3.20}$$

Following the appropriate normalization through a softmax activation, the scores are then utilized to compute the output context vector, based on the standard attention formula. The resulting context vector enriches the previous hidden state $\boldsymbol{h}_{i-1}$ and is subsequently passed to the following RNN layer.

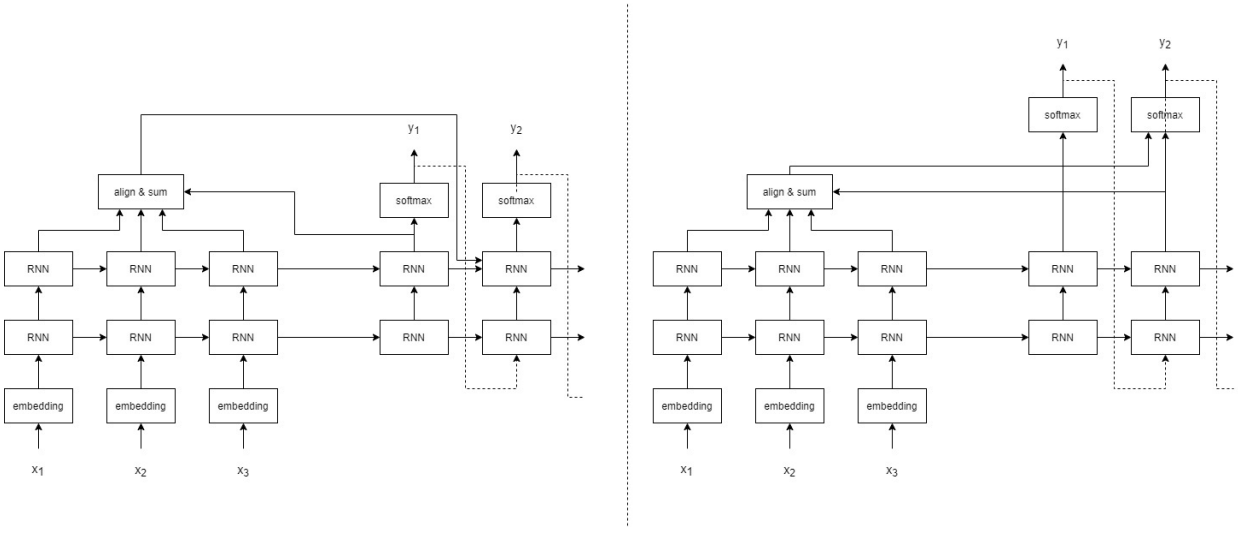

Figure 3.3: Sequence-to-sequence models with attention: Bahdanau (left) and Luong (right).

The Attention mechanism (Vaswani et al., 2017b) improved dramatically the predictive power of the sequence to sequence models. However these approaches were still facing some important issues. For example the problematic behaviour that RNNs show on

long sequences was not solved. These modules are technically unable to catch long-term dependencies that often exist in the input data. Furthermore because of the step-to-step fashion that they work, they cannot be properly parallelized. Hence both training and inference are prolonged.

3.4 Transformers

3.4.1 The Original Transformer

The Transformer model introduced a groundbreaking architectural paradigm shift when it was first unveiled at the NIPS conference in 2017 by Vaswani et al. (2017a). This innovative architecture represents a considerable divergence from the network designs that preceded it, notably because it eliminates the reliance on recurrent (or convolutional) layers previously central to addressing the temporal dynamics present in the input data. Instead, the Transformer adopts a novel approach by exclusively utilizing the Attention mechanism as its core method for processing information. Specifically, the authors of the Transformer proposed the innovative use of self-attention. This method is presented as a more sophisticated and efficient alternative to the conventional mechanisms that had been prevalent in the field up to that moment, showcasing its potential to redefine how models handle and interpret data.

In the foundational structure of Transformers, the model is designed with two core sections: an encoder component and a decoder component, paralleling the framework seen in the recurrent sequence-to-sequence (seq-to-seq) models. The encoder is charged with handling the input sequence, where it is engineered to distill a higher-level abstraction that encompasses crucial temporal dynamics, potentially extending over considerable lenghts. This is accomplished through the encoder segment being constructed from a sequence of self-attention layers, with each layer immediately followed by a Position-wise Feed-Forward Network (FFN),3.21

$$FFN(\boldsymbol{x}) = \sigma(\boldsymbol{x}W_1 + \boldsymbol{b_1})W_2 + \boldsymbol{b_2} \tag{3.21}$$

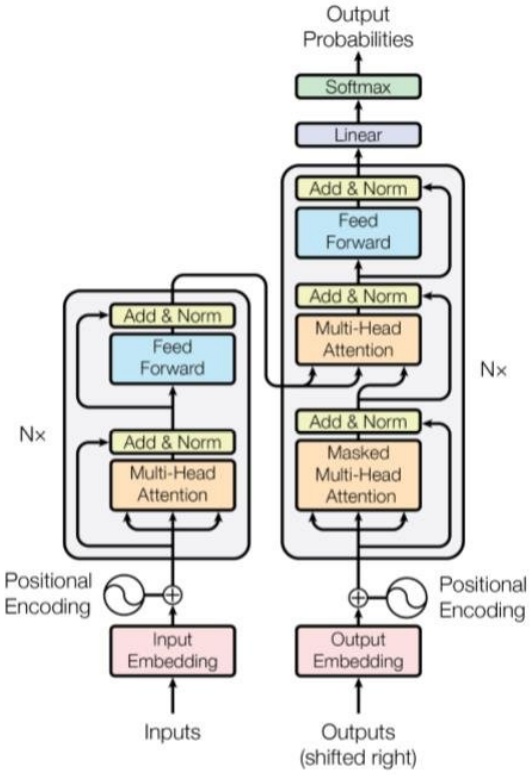

Figure 3.4: The original Transformer (Vaswani et al., 2017b)

thus enabling the encoder to proficiently learn and capture the essence of the input sequence.

On the other side, the decoder part is tasked with receiving the encoded representation of the input sequence and is then responsible for learning to regenerate the corresponding output sequence. In doing so, the decoder leverages one or more encoder-decoder attention layers, a key feature that facilitates the model's ability to continuously and effectively discern the significant relationships between the input and output sequence patterns. These specialized attention layers are strategically positioned between self-attention layers that are specific to the decoder, and are subsequently followed by Position-wise Feed-Forward Networks, thereby ensuring that the model can accurately and efficiently generate the output sequence in alignment with the encoded input.

A preliminary stage for both the encoder and decoder involves word embedding, performed in a manner similar to earlier methods we have described; the primary distinction lies in the introduction of an additional layer: the Positional Encoding layer. In subsequent sections, we will delve into and analyze key components of the Transformer mentioned

earlier in greater depth.

Attention in Transformers

As previously mentioned, attention plays a pivotal role in Transformers, being their key element. These networks utilize the attention mechanism in two distinct ways: i) as the only route of encoder-decoder interaction; ii) and in the form of Self-Attention, serving as an internal processing unit.

In the first form decoder attend the encoders finally produced representation vector for the complete set of the input modalidtes aka the whole input word sequance word-by-word. This is performed interanly at evey layer of the decoder layer intependly thought its whole depth in contrast to some earlier approaches that performed attantion only at the final layer (Luong et al., 2015).

The true novelty introduced by the Transfomer paper models is the introduction of the Self-Attention layer. Self- Attention was introduced to solve all the issues mentioned regarding the inefficty and the limitions of the previous approaches. The term refers to the usage of the Attention mechanism not to facilitate the interaction and information flow between seperate model subparts like encoder-decoder, but to process temporal dynamics inside each of these modules.

In this case all the three vector series of the Attention (q, k and v) stem for same route. In the simpplest possible form thse are identical, assuming $h = \{\boldsymbol{h}_t\}_{t=1}^{T}$ the represention vector series of interest and T its length this is translated :

$$\{\boldsymbol{q}_t\}_{t=1}^{T} = \{\boldsymbol{k}_t\}_{t=1}^{T} = \{\boldsymbol{v}_t\}_{t=1}^{T} = \{\boldsymbol{h}_t\}_{t=1}^{T} \tag{3.22}$$

As a result each sequence $\boldsymbol{h_t}$ element is now able to search the whole sequence h to find the best matching parts to extract the needed knowledge. However in practice using identical sequences leads to limitations in the representation power. For that reason the exact proposed formulation reads as :

$$\{\boldsymbol{q}_t\}_{t=1}^{T} = \{W_q\boldsymbol{h_t}\}_{t=1}^{T}, \quad \{\boldsymbol{k}_t\}_{t=1}^{T} = \{W_k\boldsymbol{h_t}\}_{t=1}^{T}, \quad \{\boldsymbol{v}_t\}_{t=1}^{T} = \{W_v\boldsymbol{h_t}\}_{t=1}^{T} \tag{3.23}$$

Where the $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ are independent projections steaming from $\boldsymbol{h}$ rather than identical to it.

The main advandage of self-Attention over recurrent or convolutionional networks is that it gives access to every part of the sequence without getting resricted in specific area. Furthermore the way that the model selects which part to interact with is fully dynamic and depends heavily on the current and input and state. These advangages improves drammaticaaly the modeling power of the network and enable and the effective processing of long sequences that was impossible before.

Additionally we are now able to run all the calculations in parallel reducing the computation time by orders of magnitude for a typical task. While recurrent layers require O(n) sequential operations to to process a seaquance of leght n, self attention allow for a O(1) lenght indepednet sequantial operations.

In the origanal paper (Vaswani et al., 2017a) attention mechanisms are implemented as a variant of the dot-product (Luong) citeluong2015effective attention. As a minor modification to the standard formulation they employed a scaling factor $\frac{1}{\sqrt{d}}$ in the scoring function that bounds gradients from reaching unpractical large values.

$$h = \{\boldsymbol{g}_i \cdot \boldsymbol{k}_i, \ \forall i \in [1, T]\} \tag{3.24}$$

$$\boldsymbol{w} = score(\boldsymbol{q}, \boldsymbol{k}) = softmax(\frac{\boldsymbol{h}}{\sqrt{d}}) \tag{3.25}$$

$$Attention(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = \sum_{i=1}^{m} w_i \boldsymbol{v}_i \tag{3.26}$$

where d the dimentionality of $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$.

Additionally, the concept of multi-head attention is introduced, wherein weighting scores and content vectors are computed in separate heads. In this architecture, Transformers do not apply attention mechanisms directly to the hidden or input layers but rather to secondary vectors that are derived from the originals via a trainable mapping. This process enhances data representation through parallel processing, enabling the model to simultaneously address different segments of the input data. This, in turn, facilitates a more nuanced and comprehensive grasp of the contextual information. When contrasted with single-head attention, the multi-head attention mechanism significantly enhances

performance, flexibility, and scalability.

In the original model, multi-head attention is realized through distinct linear transformations for each head, applied to keys (k), queries (q), and values (v). Each transformation reduces the dimensionality from the input's original size ($\mathbb{R}^{d_{model}}$) to a smaller size per head ($\mathbb{R}^{d_{head}}$), where d_{model} is the dimension of the input, and d_{head} is the dimension for each head. This mapping is achieved using unique parameter sets for keys, queries, and values as follows:

$$\boldsymbol{k}_i = \boldsymbol{k}W_i^{\boldsymbol{k}},\ \boldsymbol{q}_i = \boldsymbol{q}W_i^{\boldsymbol{q}},\ \boldsymbol{v}_i = \boldsymbol{v}W_i^{\boldsymbol{v}}, \tag{3.27}$$

for the i-th head, where $\boldsymbol{k}_i$, $\boldsymbol{q}_i$, and $\boldsymbol{v}_i$ are the transformed keys, queries, and values, respectively, and $W_i^{\boldsymbol{k}}$, $W_i^{\boldsymbol{q}}$, and $W_i^{\boldsymbol{v}}$ are the parameter matrices. The outputs from these transformations are then processed by the attention mechanism:

$$\boldsymbol{head}_i = \text{Attention}(\boldsymbol{k}_i, \boldsymbol{q}_i, \boldsymbol{v}_i), \tag{3.28}$$

and the final multi-head attention layer output is formed by concatenating the outputs from all heads and applying one more linear transformation:

$$\boldsymbol{O} = \text{Concat}(\boldsymbol{head}_1, \ldots, \boldsymbol{head}_h) \cdot W^o, \tag{3.29}$$

Positional encoding

As opposed to recurrence and convolution layers attention alone does not provide any kind of information regarding the positioning in the sequence. That raised the need of an auxiliary operation that would enrich data accordingly . To this end, authors propose the Positional Encoding (PE) mechanism. PE embeds the temporal location into the input signal at the bottoms of the encoder and decoder stacks. PE may be anything that supports the mentioned functionality either learned or fix. In the original paper authors suggest a sine and cosine functions of different frequencies.

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{3.30}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{3.31}$$

where pos the position and i the dimension. The suggested functions produce a wave like vector $\boldsymbol{pe} \in R^{dmodel}$ that is added to the original representation.

$$\boldsymbol{h}' = \boldsymbol{h} + \boldsymbol{pe} \tag{3.32}$$

3.4.2 Later Architectures and LLMs

The introduction of the original Transformer model by Vaswani et al. (Vaswani et al., 2017a) revolutionized the field of NLP and led to the development of numerous Transformer variants. Subsequent research has focused on addressing its limitations, enhancing aspects such as scalability and computational efficiency, and expanding its applicability beyond sequence-to-sequence tasks. Architectural modifications have been introduced to make Transformers more efficient and versatile. For instance, the adoption of encoder-only architectures in models like BERT (Devlin et al., 2018) has enabled breakthroughs in understanding context and semantics in natural language, transforming tasks such as sentiment analysis and question answering. Conversely, architectures like T5 (Raffel et al., 2020) and GPT (Radford et al., 2018) have excelled in generating coherent text, showcasing the model's versatility in both generating and understanding language. Efforts to improve efficiency led to innovations such as the Linformer (Wang et al., 2020), which reduces the quadratic complexity of self-attention to linear, making it feasible to process longer sequences.

The evolution of Transformer architectures culminated in the development of Large Language Models (LLMs) like GPT-3 (Brown et al., 2020) and beyond, which leverage vast amounts of data to achieve unprecedented levels of understanding and generation capabilities. These models are characterized by their massive scale, often consisting

of hundreds of billions of parameters, enabling them to develop an excellent language understanding and general knowledge. The success of LLMs is not just in their ability to perform a wide range of NLP tasks without task-specific tuning, but also in their demonstration of emergent abilities, such as reasoning and in-context learning, which were previously thought to be beyond the reach of automated systems. This shift towards larger and more capable models has sparked discussions around the ethical use of AI, the environmental impact of training such large models, and the potential for bias in their outputs. Nonetheless, the success of LLMs in a variety of complex tasks has firmly established them as a cornerstone of modern NLP research and applications.

3.4.2.1 Transformers beyond NLP

The original Transformer model was fundamentally a language technology, which remains its most prevalent application area. However, the impressive success of these models has paved the way for a multitude of diverse applications. This expansion highlights the model's flexibility and its capability to handle complex patterns in different types of data.

Vision Tranformer

A characteristic example of Transformer applications beyond Natural Language Processing tasks is Vision Transformers (ViT). These models have approached classic image processing models by substituting the popular convolutional neural networks with great success.

In the novel work (Dosovitskiy et al., 2020), the authors tackled the task by splitting the images into 16x16 patches and feeding a transformer with representations extracted from each of these. These representations are combined with a simple 1D positional embedding to inject location information. In the following stages, ViT is more or less identical to a typical Transformer encoder, consisting of a sequence of layers formed by multihead self-attention (MSA), standard normalization, and a simple MLP. To perform classification, they also use an extra learnable "classification token" as an addition to the patch sequence, the final representation of which leads to a classification head to obtain the final representation.

Given their less constrained format in comparison with CNNs, Vision Transformers are better at leveraging high volumes of training data and are usually utilized in a pretrained manner. On the negative side, this kind of model is also more prone to overfitting and often results in suboptimal outcomes when dealing with small to medium datasets. Additionally, they are slower at both training and inference compared to CNNs.

Besides the original Vision Transformer, variants have been proposed. A characteristic example is the Swin Transformer (Liu et al., 2021) that introduces a hierarchical structure allowing for efficient processing of images at multiple scales, using shifted windows rather than fix patches. Further examples include Pyramid Vision Transformer (PVT) (Wang et al., 2021), Convolutional Vision Transformers (CvT) (Wu et al., 2021), and many more.

Video Tranformers

Transformers have extended their utility beyond static image analysis to encompass video processing, capitalizing on their adeptness in managing dynamic and temporal information. The emergence of the Video Vision Transformer (ViVit), as introduced by Arnab et al. (Arnab et al., 2021), marks a significant progression, demonstrating novel methodologies for video analysis. ViVit conceptualizes videos as sequences of 3D patches, offering various models that either holistically capture both spatial and temporal dynamics or amalgamate separate spatial and temporal self-attention mechanisms in different configurations.

Subsequent to ViVit, innovations such as the TimeSformer by Bertasius et al. (Bertasius et al., 2021), MViT by Fan et al. (Fan et al., 2021), and the Video Swin Transformer by Liu et al. (Liu et al., 2021) have further propelled the domain of video analysis. These models enhance the foundational capabilities introduced by ViVit, delivering refined strategies for interpreting video content. Finally, transformers have been effectively employed in conjunction with various static image processing architectures, treating the task as a generic sequential data modeling challenge without specific constraints. The widespread success of these implementations underscores the significant potential of transformer networks in modeling both spatial and temporal structures.

Chapter 4

Sign Language Translation

Sign languages are formal, independent languages that employ purely non-verbal means of expression. The field of Automatic Sign Language Processing, especially Sign Language Translation, presents a significant research avenue with deep societal impacts. Importantly, end-to-end SLT improves communication between the Hard-of-Hearing (HoH) community and hearing individuals, thereby enhancing their social integration and opening up more opportunities. Unfortunetly despite its significance, this research area is still in its infancy.

The application of Machine Learning to sign language issues covers a broad spectrum, yet the majority of existing literature does not focus on the actual translation of sign language but rather on related aspects like sign recognition. In contrast to the common misconception sign languages are not extensions of spoken languages. Each national sign language has its own unique grammar, syntax, and vocabulary, without any direct connection to the spoken language of the same country. As a result, a true SLT system must recognize the visual patterns of sign language, decode their linguistic elements, and translate them into a spoken language. This represents a substantial technical challenge.

In this chapter, after covering core aspects of sign languages and relevant Machine Learning approaches, we propose an end-to-end SLT model utilizing a novel doubly stochastic transformer. Our approach integrates stochastic local linear competition with a Gaussian variational treatment of the trainable parameters. This model has enabled us to achieve state-of-the-art results on the most extensively used dataset in the field. Furthermore expanding our research beyond the standard and convenient benchmarks, using an alteration of our model, we have developed the first multi-topic, end-to-end model for Greek SLT, an application of greater real-world value. This advancement was made possible through the compilation of original data from formal elementary school materials.

4.1 Sign Languages

A Sign Language (SL) mainly uses manual (e.g., hand shapes, movement of the hands, arms or body), and facial expressions to fluidly express a speaker's thoughts and constitutes the main communication means for deaf people. National SLs, being the mother tongues of deaf SL users, are an important part of the European and the world cultural diversity. For the deaf the access to SL communication is essential because it facilitates access to equal education, employment, information or to health services (Dreuw et al., 2010). In Europe, there are 30 official SLs and more than 750000 deaf SL users, while only 12000 interpreters. This shortage undermines the right to equal education and health services and often endangers the lives of deaf people (Murray, 2013).

There is an intrinsic motivation in human populations to communicate with high level of information-density. In linguistics (verbal communication) the principal communication channel is auditory (speech). However, in cases where the speech modality is not applicable, an alternative communication channel is used; the visual (manual) channel. Sign languages utilize the visual channel and are officially considered to be a verbal communication due to their vocabulary, grammar and other linguistic structures (Sandler &Lillo-Martin, 2006). Sign languages abide the necessary requirements and are viewed as full-fledged languages (Bellugi &Fischer, 1972). A typical characteristic of natural languages is that they emerge and evolve over time without meticulous planning.

In a community with many human individuals that use the visual-manual channel for communication, it is eminent the development of language properties within a few years. The language becomes structured, fluent, synchronized and possible of expressing any conceivable concept. Being a natural language, sign languages inherit traits specific to the community thus are not universal even though there are similarities among them. A prime example is the Nicaraguan Sign Language (Senghas &Coppola, 2001). An interesting case of sign language that was developed by deaf children in various schools in western Nicaragua in the 1970s. It was spontaneously developed among humans (children) with hearing loss and offered a unique opportunity to linguists to study the birth of a new sign

language.

An important part of sign language is the non-verbal communication elements. For instance, gestures, body posture, facial expressions, eye movements and speed, intensity and size of signs (Wang et al., 2003). These elements if used separately are considered body language. However, we must not confuse sign language with body language, there is an important distinction between the two. Body language is considered the bundle of various non-verbal communication elements that are used individually while in sign language are utilized as tools for amplifying the language. A direct equivalent with spoken languages is the paralanguage (e.g. rhythm, intonation, pitch, volume). These elements are of principle importance in both types of natural languages for enriching the language and allow the speaker to convey the desired message.

Culture and Linguistic Challenges

Sign Languages is a complex and challenging area that requires multilevel approach in order to be well modeled and fully understood. Therefore, a processing system aiming to achieve a novel result, needs to take in account all the linguistic characteristic and identify the major obstacles coming from the SL culture and linguistic uniqueness(Bragg et al., 2019).

Variety in fluency is an example of the challenges that a researcher needs to overcome. Some people have acquired hearing or speech problems during a later stage of their life, and which means little practical experience communicating via SL. On the other hand, people born with acoustic problems and lifelong SL users with potentially excellent skills on their mother tongue(Mayberry &Kluender, 2018). However even deaf and mute born children who luck of special education may also have reduced fluency. Variance in fluency practically means variance in the signing speed, the vocabulary size and the number of mistakes, all of them parameters with critical effect on the automatic translation.(Bragg et al., 2019).

The academics distinguish two major categories of deaf children: Deaf children of deaf parents (DCDP) and deaf children of hearing parents (DCHP). It is estimated that from

the total population of deaf people, only 5% are DCDP. This puts in a disadvantage most of the deaf children, since the parents are not typically equipped to fully understand the needs of the children. DCDP are the native speakers of the language since the primary communication exchange at home is SL. A hearing parent has no knowledge and expertise regarding SL, thus cannot efficiently understand the needs of their child. **Deaf people are visual**.

Take for example the parents of a deaf child answering the front door of their house. A DCHP child cannot understand how his/her parents know when a visitor is outside the door. This might result on behavior of a child opening the door and becoming upset that nobody is there on the other side. This is an example of the information deaf people cannot receive due to their disability which is not inherent to their parents. Moreover, DCHP children tend to receive the word to word correspondence of the parent's native language to their sign language avoiding the grammatical structure of a sentence. This is miles different from the actual conversation observed by a native/fluent speaker. The ordering of signs to create a sentence is much different from their spoken language equivalent. Typically, fluent signers are hearing children of deaf parents (HCDP). Hence, when performing research on SL, researchers prefer the assistance of DCDP or HCDP for their point of reference.

Sign vocabularies can also hide a few unexpected kinds of obstacles. Similarly, to common languages it does exist a vocabulary that includes the core sign and gestures for the corresponding SL but is not very strictly defined. The gestures and their meanings have a more abstract and loose meaning and it is practically impossible to measure the exact size of the vocabulary. Furthermore, SLs are not limited to the language specific signs, further auxiliary gestures such as classifiers also exist and have an important role (Bragg et al., 2019). Additionally, the wide internal diversity of each sign language should be also mentioned. Even more than the spoken languages the vocabulary and the singing style of them is heavily influenced by the geographic location and the social subgroups of the users.

4.2 Automatic Sign Language Processing

The applications of Machine Learning methods in Sign Language Processing(SLP) are extensive, encompassing a diverse range of methodologies. All methods share the common objective of assisting the deaf and hard-of-hearing (HoH) community; furthermore, they frequently encounter similar technical aspects and challenges. These challenges include navigating the complex and multimodal nature of Sign Language and the necessity for high-quality data, which are challenging to collect. We analyze these technical aspects and present significant data collections and prior works. Our focus is on the task of sign language recognition, which is the most frequently occurring case, and, more importantly, on sign language translation; this is our principal objective and the most comprehensive application.

4.2.1 Sign Language Data

In the existing literature, a variety of datasets are available for sign language-related tasks. Among these, the DGS Kinect 40 dataset includes 40 signs from Deutsche Gebärdensprache (German Sign Language, DGS) (Ong et al., 2012; Elliott et al., 2011). The GSL 982 dataset features 982 signs from Greek Sign Language (GSL), with 5 examples of each sign performed by a single native signer, resulting in a total of 4910 samples. The weRWTH-PHOENIX dataset focuses on Word-Level American Sign Language (WLASL) with more than 2000 words performed by over 100 signers (Li et al., 2020). The SIGNUM dataset comprises 450 German language signs (Agris &Kraiss, 2010), and the American Sign Language Lexicon Video Dataset includes more than 3,300 ASL signs, each produced by 1-6 native ASL signers, totaling almost 9,800 tokens (Athitsos et al., 2008). Additionally, datasets for Polish Sign Language (Oszust &Wysocki, 2013, 2012), LSA64 for Argentinian Sign Language (Ronchetti et al., 2016), AUTSL for Turkish Sign Language with 26 signs performed by 43 signers (Sincan &Keles, 2020), and BosphorusSign22k (Camgöz et al., 2016; Özdemir et al., 2020) are mentioned. The MSRGesture3D dataset includes 12 dynamic ASL gestures performed by 10 people, with a total of 336 files (Kurakin et al., 2012; Wang

et al., 2012). Other cases include Devisign, featuring 23 different gestures captured at 30 fps and 320 by 240 pixels with a Sony Handycam (Chai et al., 2015); an Indian Sign Language dataset (Nandy et al., 2010); the Purdue dataset, which contains various ASL signs and narratives produced under controlled conditions by 14 fluent Deaf ASL signers (Martínez et al., 2002); Cuny ASL (Lu &Huenerfauth, 2014); and the SignsWorld Atlas, a benchmark Arabic Sign Language database (Shohieb et al., 2015)

The availability of SL-related datasets is clearly considerable. However, only a select few are suitable for the most critical application of SL processing, namely end-to-end SLT. A dataset is suitable for SLT model training if it possesses two crucial characteristics: i) it includes SL videos paired with corresponding translations in a formal spoken language, and ii) the video-text pairs comprise complete and syntactically correct sentences/phrases of adequate length.

In this context, Phoenix2014T (Camgoz et al., 2018) dataset has become one of the most extensively studied datasets for this purpose. It features weather news performed in German Sign Language, and includes both Gloss annotation, an auxiliary pseudo-text form of Sign Language, and text translations. The inclusion of Gloss annotation, combined with its dense single-topic vocabulary, renders the Phoenix2014T dataset more conducive to deep learning and has thus attracted significant attention.

Other notable datasets in this domain include SWISSTXT-NEWS (Camgöz et al., 2021) and VRT-NEWS (Camgöz et al., 2021), which cover a broader range of topics. These datasets are composed of TV-news data in German and Flemish Spoken and Sign Languages. However, the results of any end-to-end SLT attempts on these datasets have been disappointingly low, as they have failed to achieve even remotely acceptable translation quality. A recent important addition to this field was a project published by the BBC (Albanie et al., 2021), consisting of a particularly large number of phrases in British Sign Language and English spoken language. The potential of this new dataset is especially high, mainly due to its extensive size. Nonetheless, end-to-end SLT results of deep learning models trained on this dataset have yet to appear in the related literature. Further examples are the CSL-Daily on Chinese SL with properties similar to Phoenix2014T and

the American-SL dataset How2Sign (Duarte et al., 2021) with good size and quality but lower reported results. A very recent and important addition is the OpenASL dataset (Shi et al., 2022), published in late 2022, covering 300h of American SL videos collected from online videos and demonstrating respectable results.

4.2.2 Communication Channels and Features

As previously stated, sign language is a structured form of visual motions which are used for communication. In order to convey information, sign language utilizes the visual channel and entails the usage of hand shapes, number of hands, upper body locations, movements of the hands and arms, distance between the hands, facial features, head movements and self-contact. Feature extraction from images is of utmost importance for tackling SLP tasks, hence the scientific community's active interest on detecting and analyzing related features in an a efficient and scalable manner.

Hand Shape

Hand Shapes are significantly important for all the signed languages and therefore reasonably was one of the very first features that have been studied. Typically extraction methodologies around this type of features aim to detect key gestures of the hand that can give direct value to entire system. Hand gestures can be either static poses or short dynamic movements.With constrains on the linguistically relevant hand-shapes been investigated by (Cui et al., 2017; Dilsizian et al., 2014; Ding &Martinez, 2009, 2007; Koller et al., 2016a; Ricco &Tomasi, 2009; Thangali et al., 2011). Improvements of the hand-shape recognition accuracy has been achieved by leveraging phonological constraints on start and en hand-shape co-occurrence (Dilsizian et al., 2014; Thangali et al., 2011).

Several hand shape recognition methods have been proposed by the community (Athitsos &Sclaroff, 2001, 2003; Heap &Hogg, 1996; Isaacs &Foo, 2004; Lu et al., 2003; Tompson et al., 2014; Vogler &Metaxas, 2004) using a variety of different methodologies. The majority of the recent approaches utilizing convolutional neural networks have been able to achieve state-of-the-art performance on hand-shape recognition given a large dataset

(Koller et al., 2016a; Hussain et al., 2017). Additionally to them, anatomy aware techniques that employ the convexity hull algorithm seems also to performs well in recognising a set of hand shapes in 2D visual signals. (Nikam &Ambekar, 2016)

Beside the extraction of features related to specific key poses and shapes, some attempts tried to employ more descriptive and detailed properties. Properties that may not be directly correlated with the target but contribute by helping the model to understand better the visual figure of the hand. For example analysing and and detecting the finger figures gave positive results in older attempts(Ravikiran et al., 2009). Modern techniques using more advanced equipment like multi-camera systems evolved this kind of features and make them much more sophisticated. Such a model was proposed by Simon et al. (2017), who designed accurate detectors for hand key points like finger joints.

Hand Motion Trajectories

Apart of their shape, hands movements and their exact positioning over the body figure are also equally important. Hand and arm movements are an integral part of the main signing stream while the positioning can affects the expressed meaning dramatically. Several researchers (Dilsizian et al., 2016; Han et al., 2009; Pu et al., 2016b) show that tracking the hand motion and utilizing the inferred trajectories can be used as intermediate step towards a more accurate sign recognition. A hand motion trajectory can be 2D or 3D and it seems that both captures can be beneficial to an SRL system.

The first case has been studied by researchers (Ding &Martinez, 2009, 2007) who exhibit the efficient combination of motion trajectories with typical hand and face features used in ASR. 2D tracking is often based on skin color detection technique (Kolkur et al., 2017) supported by Gaussian mixture models for an accurate representation of the color distribution (Hammer &Beyerer, 2013)(Hammer et al., 2016), while the trajectory is taken using the body figure as reference shape. Deep learning models from the wider are of object localisation like the Faster R-CNN(Ren et al., 2015) can also trained to localise the hands with great accuracy (Liao et al., 2019). The advantages of the two dimensional version is undeniably its simplicity and the fact that nothing more than a simple rgb

camera is need to collect the data.

On the other hand more recent advancements on 3D motion trajectories provide a more comprehensive and rich representation of the available information (Dilsizian et al., 2016). Equipment like the Microft Kinect provide these signals directly (Dilsizian et al., 2016) utilizes the 3D motion trajectories in ASR and demonstrates their importance.

Facial Expressions

Signed languages are mainly focused on hand movements, however hand related features rarely have enough expressive power. Facial expressions(FEs) are probably the most crucial auxiliary feature that supports well the hand signed sequences. It is interesting that the meaning of a hand shape or gesture can sometimes be significantly or even entirely changed because of a mild variations of the FE (Elliott &Jacobs, 2013). More specifically FEs affects the transmitted signs via two discrete information pipes (McCullough et al., 2005) revealing emotions and directly adding grammatical concepts.

Our face is indisputably the main means of expressing our emotional condition. Similarly to any other natural verbal language SL is highly correlated with emotion and therefore the information that we can extract from the face can be an extremely valuable additional ally in SLP (Goldstein et al., 2000; Elliott &Jacobs, 2013; McCullough et al., 2005). Emotion Recognition is by itself an active and very interest research area of Artificial Intelligence and Computer Vision. Several labs have worked and are still working on the topic giving exceptional results that are constantly improving. While earlier work had focused on support vector machines and other techniques, currently Deep Learning is the dominant method. Just like many other vision problems the current state the art models are variations of well known convolutional DNNs such as VGG16 and Inception (Tzirakis et al., 2017)(Guo et al., 2018). These models can be directly applied to the signing data and provide the needed emotional features.

Beside the expression of emotional information the second and even more important role that the FEs play, in the sign language, is the direct addition of grammatical aspects to the hand signing. There are several FEs that correspond to specific grammatical or syntactical

transformations of the linguistic meaning, those expressions are called Grammatical Facial Expressions (GFEs)(Freitas et al., 2017). Their importance can be easier understood if we consider that they can convert regular sentences to questions or negations and highlight the important information in a sentence. GFEs are an integral part of SL and any complete recognition should include them in its framework (Guerra et al., 2018). The methodologies for detecting GFEs are similar to what was previously mentioned based on variotons of neural networks (Xu; Walawalkar, 2017).

Despite the undeniable importance of emtotion and GFEs face contribution is not limited to them. The analysis of lips and mouth shapes can be another very helpful supporting source with a wider role. A relevant paper examined methodologies for detecting SL related mouth shapes (Koller et al., 2015) . Focusing on the detection of core phoneme they suggest the usage of CNN models combined with hidden markov models for an accurate classification of 39 classes. Facial features extraction methods can also have a more indirect approach focusing in other auxiliary tools such as facial landmarks (Walawalkar, 2017).

Body Pose and Skeletal Features

Sign language is technically using the whole upper part of the human body and therefore parts like shoulders, chest and generally the overall body pose are also transmitting information and they are highly valuable for a SL recognition system. Similarly to hand shapes body pose features may have different forms eg. the recognition of specific poses and movements or tracking of specific body parts. The most comprehensive way is the estimation of a full body skeletal model that can describes well the body figure and the most important motions.

The current bibliography covers a wide spectrum of topics on pose estimation as an isolated computer vision problem as well as for its usage in sign language recognition. (Cao et al., 2017). Many novel models are employing them as a part of their pipelines and sometimes as the exclusive source of feature source for the main model(Bilal et al., 2011). Recent hardware technological advancements like Microsoft Kinect and 3d cameras

evolved the pose estimation taking to an other level (Monir et al., 2012) since their able to provide specialised and highly valuable data. However regular 2d rgb video can also be a reliable feature source. Several extraction techniques had been tried over the last decades like Specialized Mappings Architectures (Rosales &Sclaroff, 2002) and markov chain monte carlo algorithms (Lee &Cohen, 2004) with good accuracy. Recent works are mainly based on deep learning methodologies and manage to build very accurate skeletal representations of the body shapes. More specifically architectures like VGG-19 network are employed in order to joins joins and other key body parts. A more complete skeletal representation may be a following step built using the detected points of interest. A possible issue of 2d video is that many of the key joints may be hidden from the camera's point of view resulting to wrong predictions. The problem can be partially solved by introducing confidences scores on the predictions that suggest if one should be consider or not.

4.2.3 Sign Language Recognition

Sign Language Recognition (SLR) is the automatic procedure that targets on recognising discrete sign from sign language videos The motivation behind is to bridge the gap of communication between the deaf and non-deaf community. Several ideas for products that recognize gestures have been investigated, e.g., using gloves or wristbands that measure electrical activity. Unfortunately, they all remain prototypes mainly due to their obtrusiveness that renders them infeasible to be widely employed for Sign Language Recognition (SLR). In contrast, computer vision approaches are very attractive for SLR because (a) cameras are cheap and embedded in many devices (b) provide rich information including body and face and (c) modeling of video data is becoming feasible using the latest machine learning methods.

Static

The simplest form of sign language recognition is the recognition of a predefined set of static hand poses in 2D image data. This set can be a language specific dictionary of signs or supporting shapes like finger spelling and classifiers. Formulated as an image

recognition problem it has benefited from the recent developments of the wider area of computer vision with very reliable results. Furthermore, while not trivial the specific task is less complicated than the continuous translation and even early attempts performed quite well.

Model formulation based on Support Vector Machines have been used in several independent studies and showed that they can outperform several other methodologies such as k-NN and Naive Bayes (Cheok et al., 2019). Applied on a dataset of approximately 1800 gray scale Webcam images of hand gestures an SVM algorithm with linear kernel was able to achieve 82.3% accuracy on the classification of 25 signs and 99.2% on the subset of 12 signs (Kurdyumov et al., 2011). The introduction of Scale Invariant Features Transform SVMs has helped to achieve 99% accuracy on the Arabic Sign language recognition task (Tharwat et al., 2015).

Simple fully connected neural networks had also been investigated. Typically, researchers paired NNs with rigorous pre-processing and feature extraction processes to increase the predictive power. A method that used a NN of just 1 hidden layer combined with a continuous wavelet transform managed to achieve more than 99% accuracy on classifying 24 finger spelling letters (Isaacs &Foo, 2004). An almost identical approach accomplishes 94% on the recognition of Persian SL (Karami et al., 2011). Although the models have been able to achieve notable accuracy scores, all of them were applied to clean data that were captured in an ideal setting.

More recent projects employed modern sophisticated deep convolutional architectures and managed to extend the accurate classification for wider and more realistic signing data sets (Koller et al., 2016a; Hussain et al., 2017; Ameen &Vadera, 2017; Li et al., 2019). An example is a paper by Ameen and Vadera (Ameen &Vadera, 2017) who worked on a more "in the wild" dataset and managed to achieved 90% accuracy on the American SL prediction task. Well known computer vision architectures, that have shown state-of-the-art performance in other image recognition tasks, are now being re-purposed to solve SLR tasks(Simonyan &Zisserman, 2014). In 2017 a study comparing several architectures found VGG16 to outperform most of the alternatives (Quiroga et al., 2017) on LSA16

and RWTH-PHOENIX-Weather handshape datasets. Alternatively, custom-made Deep Convolutional NNs have been proposed (Oyedotun &Khashman, 2017).

Dynamic

Static hand signs have limited use for real life applications. A more useful and challenging task is the recognition of dynamic gestures. Dynamic hand gestures are defined as a sequence of hand poses and motions of short duration entailing a single linguistic expression. Moreover, gestures can also include motions from the whole human body and face. Gestures can act as the words that form the vocabulary of a sign language or as short phrases and expressions. Designing efficient dynamic gesture prediction models can provide the foundations for the design of accurate sign language translation systems.

Over the last decades the scientific community had performed numerous studies on the dynamic gesture classification by employing a variety of dynamic modeling techniques such as Dynamic Time Warping, Finite State Machines, Kalman Filtering etc. All these techniques are applied in conjunction with pre-processing and feature extraction processes such as isolated signs and 2D motion trajectories. A study on single signer recognition on 984 signs (Cooper et al., 2011) has shown 71.4% top-1 accuracy performance. A study on multiple signers on 1000 signs (Wang et al., 2016) has shown 70.9% accuracy performance. Apart of the mentioned methods the remaining literature is heavily based on two techniques, Hidden Markov Models, a historical powerhouse in time series analysis, and Deep Learning which has shown state-of-the-art performance (Cheok et al., 2019; Ahmed et al., 2016; Elmezain et al., 2008).

HMMs seems to be one of the most popular and successful methodologies over the years. The first attempts on the applications of HMMs started in the early 90s with projects like the remarkable paper of Thad Starner (Starner, 1995) in which using HMMs have managed to build one of the first a hand gestures recognition system. The system was based on features derived from the positioning of the hand in the 2D space and the angles of the hand axis. In order to make the tracking more accurate during the data collection process the singers wore a pair of colored gloves. The attempt is considered

successful since they have managed to achieve a 95% accuracy performance on predicting 395 sequences. These were derived from five-word length combinations of a 40-word subset of the American sign language vocabulary.

There is an extensive line of work for the task of dynamic gesture recognition using HMMs. Initially the research was focused exclusively on American and major European languages with some notable work on less studied languages (Cheok et al., 2019; Al-Rousan et al., 2009). A prime example is the 2009 publication (Al-Rousan et al., 2009) on the Standard Arabic Sign Language. In this work, the authors manage to classify 30 isolated words from the Arabic SL in both signer dependent and independent modes. In order to find the performance in realistic conditions the authors used information from 2D videos taken in various lightning backgrounds. Moreover, the usage of gloves or any other kind of auxiliary tool was excluded. The authors used a pre-processing step consisting of discrete cosine transformation with zonal encoding applied on each frame. This resulted in a reduced state space of interest that was represented by an element feature vector of size 50. The HMM was constructed and trained on the sequences of these feature vectors. When trained on specific signers it achieved 96.6% on offline data and 93.8% on an online test, while for the signer-independent case the corresponding scores were 94.2% and 90.6% respectively.

Other variations of HMMs for dynamic gesture recognition include Parametric HMM (Wilson &Bobick, 1999), factorial HMMs (Ghahramani &Jordan, 1996; Brand et al., 1997) and Parallel HMMs (Vogler &Metaxas, 1999). While comparisons of different approaches can be seen in (Liu &Lovell, 2003) that compared topological structures (left-right and full connected) exhibiting minimal change on classifying performance; and (Brand et al., 1997) that compared linked HMMs and couple HMMs exhibiting less initial value sensitivity to the former. Finally, HMMs can be combined with NNs (Koller et al., 2016b,a).

Dynamic Gesture Classification, as with several other computer vision tasks, is now being mostly tackled with Deep Learning (DL) techniques. The current state-of-the-art architectures are heavily dependent on convolution and/or recurrent layers in order to capture the spatio-temporal nature of the data. The formulation that includes Recurrent

Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) could have several forms. One approach includes the training of a CNN model separately on the spatial features and then use the prediction as the input for the subsequent RNN. That was applied by (Masood et al., 2018) where for each gesture video they labeled one by one all the frames and trained an Inception model (Szegedy et al., 2015) to predict which is the gesture that each image/frame belongs to. Subsequently, they trained an RNN model based on the sequences of these predictions. Additionally, they trained the RNN not on the prediction of the Inception model but on an intermediate layer. The latter was shown to produce an improved accuracy of 95% accuracy performance over the 80% accuracy of the former approach.

While two phase training session like the previous ones could be very successful it is often desirable models to perform and learn in a more autonomous and end-end way. 3D convolutional Neural Networks is an architecture that allowed these properties (Molchanov et al., 2015; Liang et al., 2018; Huang et al., 2015). In contrast to the more commonly used 2D CNNs the 3D version has the capacity to extract spatial-temporal information out of a video signal. They act as a simple expansion of the convolution layer where 3d kernels map 3d spaces to feature maps, with the additional dimension being time (Pu et al., 2016a; Liang et al., 2018). The layer introduces a larger number of parameters that in turn increase the computational cost. To moderate the problem researches tried to initialize the weights of a 3D CNN by using 2D weights learned from ImageNet, while other propose a 3D CNN that factorizes the 3D convolutional kernel learning as a sequential process of learning 2D spatial and 1D temporal kernels in different layers.

A paper from Huang et al. in 2015 (Huang et al., 2015) is one of the first and characteristic attempts to employ 3D-CNN for the recognition of isolated signs. In order to perform their experiments, they collected a new dataset using the Microsoft Kinect that additionally to RGB video data has provided them with depth and skeletal information. The data were inputted to the model in the form of nine stacked frames of 48x68 pixels. The full architecture included 8 layers including three 3D-Convectional layers with kernel sizes 7x7x5, 7x7x3 and 5x5x3 respectively and 50 filters each. Between the convolution

layers they included a 2x2 sub-sampling layer and at the end two dense layers. The model was evaluated based on the recognition of 25 different words-sings. They have achieved 88.5% accuracy when applied on gray scale image and 94.6% on all the five different feature maps (R,G,B,depth,skeletal). In a similar vein (Liang et al., 2018) published in 2018 a 3D-CNN model. Their architecture starts with three 3D Convolutions of different kernel size (5x5x5 ,5x5x5 4x4x4 and 16,32 and 48 feature maps respectively) all of them activated with a ReLU function. Similar with the latter they include a pooling layer between the 3D convolutions. For the final part of the model they propose three dense layers. Their model was trained with infrared and contour stream collected via a Microsoft Kinect camera. They also include the tracking information of the hands as auxiliary input to the network.

A recent paper (Liao et al., 2019) combines 3D convolution with bidirectional LSTM exploring deeper architectures. They propose a model called B3D ResNet which was able to capture visual features in various temporal windows from video. The full architecture includes 17 3D convolutional layers the first one with a 7x7x7 kernel size and the rest 3x3x3. Excluding the first layer convolutions are uniformly organized in four groups, each one of those with internal residual connections and a pooling layer. Subsequently, the formulation includes two Bidirectional-LSTM layers and two dense layers. The convolutional part of the architecture has the capacity to extract spatio-temporal dependencies which are in turn processed by the Bi-LSTM in order to capture long term dependencies and more accurately predict the final labels. B3D Resnet architecture includes one more auxiliary NN. Specifically, a version of faster R-CNN(Ren et al., 2015) is employed in order to isolate the hand shapes in the RGB frames. After the hand position is localized it can be used as a second input to the main model, accompanied with the initial RGB video which is the main source. The whole model was evaluated on the SLR and DEVISIGN-D datasets. When the B3D ResNet trained and applied only on the initial video it managed to achieve 50.2% and 43.7% accuracy scores. On the other hand, when hand localizations were used to scores jumped to 86.9% and 89.8% respectively surpassing other approaches.

Many researchers manage to build accurate classifiers for static hand signs and short dynamic gestures that represent specific words. Nevertheless, a real-life application of such

a system must include models that are able to recognize all the transmitted information in a continuous and realistic environment. The simplest solution is the continued gesture detection. Instead of detecting single words in short videos of predefined duration, models should be trained to detect and localize temporally specific gestures one by one in longer videos. In that way they can export sequences of signs that build a greater meaning. For instance, in the previously mentioned project (Liang et al., 2018) in order to deal with real time streams they employed temporal segmentation.

Sign language gestures are loosely consisted of three phases preparation, nucleus and retraction (Zhu et al., 2016). Nucleus includes all the information while the other two more or less similar in all the gestures and do not provide any linguistic information. Considering this structure, the authors have implemented a simple 2-layer NN with binary output that learned to mark each frame with the comprehensive label. In that way they managed to detect the beginning and the end of each word thus making the transition from the isolate gesture recognition to continuous gesture recognition smoother.

Finally, DNNs can also be applied to non-video information. For instance, the trajectories of just four skeletal joints was the only feature that (Liu et al., 2016) used for their deep learning model. Their architecture included primarily an LSTM and they have managed to show that DNNs outperform all other non NN-based alternatives in the recognition of 500 Chinese SL words by 50 signers.

4.2.4 Sign Language Translation

The field of automatic sign language recognition has garnered considerable attention in academic research. Nonetheless, the majority of these studies have concentrated on identifying hand gestures or upper body movements. Although these studies hold substantial practical importance, they often fail to provide a comprehensive overview of the subject.

Sign Language Translation (SLT) represents an emerging area that aims to offer a more integrated perspective. Technically, SLT merges elements of machine translation and computer vision, with its fundamental goal being to generate translations that are both

syntactically and grammatically accurate, deriving this complex linguistic information solely from visual inputs.

Ideally, SLT models process the input SL videos in a complete end-to-end manner, learning to address all spatial, temporal, and linguistic aspects and produce the corresponding translation. This forms a particularly challenging task, almost impossible from a technical perspective. The complexity inherent in modern Deep Learning models necessitates an enormous amount of data, which, as mentioned before, is far from readily available or easy to collect. Additionally, the execution of these models requires very high-end hardware. This leads to some alterations of the core task to make it feasible from a technical perspective.

To render the SLT task feasible, various alterations and techniques are employed. Typically, this includes the use of transfer learning methodologies utilizing pre-trained network parts for specific purposes. Additionally, auxiliary data are usually employed to assist and guide the training procedure. The most frequently occurring example is the use of so-called gloss annotations. As previously mentioned, glosses are a type of pseudolanguage corresponding to a kind of text form of a sign language. Technically, a gloss represents a specific sign in a discretized form. However, due to the fluid and multi-channel nature of Sign Language, glosses are far from perfectly mirroring any signed content, especially at the sentence level. Additionally, for this reason, gloss annotation is particularly laborious and requires deep expertise and experience.

Methodologies adopted in addressing the SLT challenge are often organized into different subgroups, with key differences often revolving around the use of gloss annotations. One prevalent method incorporates these gloss labels as an intermediary conduit, bridging the gap between the visual representation of sign language and its translation. Conversely, some strategies focus on direct translation from sign language to text, typically with limited or no dependence on gloss annotations. The subsequent sections will expound on the principal classifications within the SLT domain.

Gloss to Text

The simplest SLT type is the Gloss to Text (G2T) translation. In this case, we do not perform true Sign Language Translation but rather a theoretical modeling, assuming perfect recognition of the visual signals. While not directly useful for real-world applications, G2T provides an ideal laboratory environment. These conditions allow for isolated experimentation on the linguistic part of SLT, enabling procedures like hyper-parameter tuning and architecture investigations. Additionally, due to its simplicity, G2T is a common and fast benchmark for the scientific community. However, it's important to note that glosses as an incomplete language channel often fail to perfectly express the information.

Sign to Gloss to Text

The Sign to Gloss to Text (S2G2T) method is one of the most acclaimed approaches in the task, being the first to attain notable results. In this method, Gloss acts as the intermediary link between video and text, structured in a dual-stage process. The first stage of a typical S2G2T design involves a recognition system that extracts the gloss sequence from a video input. This predicted gloss sequence subsequently becomes the input for the ensuing Gloss to Text (G2T) network. Depending on the variant, the G2T part may be trained using either the predicted gloss sequence or the actual ground truth. The key disadvantages of this kind of approach include the need for gloss annotation for any given dataset and limitations that involve the usage of those. This methodology has spawned a wide range of works and papers, introducing highly successful techniques and various architectures.

Sign to Text

Sign to Text (S2T) SLT architectures translate sign language videos directly into spoken language text and are undoubtedly the most comprehensive. This holistic end-to-end approach makes S2T a particularly challenging task, as it simultaneously captures sequential, visual, and linguistic signals. However, because S2T does not depend on glosses as an intermediate step, it theoretically has the potential to produce results superior to G2T,

since it has direct access to the original source of information and, consequently, to all types of auxiliary signals, as previously discussed. Due to the complexity of this approach, it often relies on pretrained external parts to process the spatial dynamics of the input modality, referred to as Feature Extractor.

Previous RNN based approaches

Neural Sign Language Translation (Cihan Camgoz et al., 2018) was the first study to achieve high-quality translations on a significant dataset, integrating advancements from NLP and computer vision. It introduced a seq-to-seq framework utilizing an encoder-decoder pair of recurrent neural networks, tailored for sign language translation (SLT) to process video inputs instead of textual sentences.

The research explored three methodologies: Gloss-to-Text (G2T), Sign-to-Gloss-to-Text (S2G2T), and Direct Sign-to-Text (S2T), starting with G2T as the foundational approach. Through comprehensive testing, the authors optimized a 4-layer GRU network with an embedding size of 1000 and the Luong attention mechanism, yielding BLEU-4 scores of 20.19 and 19.26 on the development and test datasets, respectively (Cihan Camgoz et al., 2018).

Advancing to S2G2T, upon fine-tuning the G2T setup, the team tested the model using outputs from a Sign Language Recognition (SLR) system as input, achieving BLEU-4 scores of 17.86/17.79 (dev/test). By training the model with these gloss estimations, they further improved the performance by approximately 0.5 BLEU-4 (Cihan Camgoz et al., 2018).

In their S2T experiments, the authors aimed to generate spoken language directly from sign videos without an intermediate gloss representation, incorporating a CNN (ALEXNET) with the seq-to-seq architecture to analyze video frames. However, this approach yielded BLEU-4 scores below 10, indicating the challenges of direct translation without glosses. Related works in this domain include Partaourides et al. (2020), Xiao et al. (2020), and Bendarkar et al. (2021).

Sign Language Transformers

Transformers emerged as a potent solution for the S2T challenge, surpassing gloss-dependent models by about 10%. The adapted transformer for S2T involved a conventional encoder-decoder architecture with a spatial embedding module for extracting spatial information on a frame-by-frame basis, similarly to (Cihan Camgoz et al., 2018).

The study (Camgöz et al., 2020) achieved significant advancements using glosses not as an intermediary but as auxiliary supervision. They proposed a 3-layer transformer designed to predict glosses and text jointly. The encoder's output, in addition to feeding the decoder, facilitated sign language recognition via a CTC loss. The decoder followed the seq-to-seq paradigm for translation, recording BLEU-4 scores of 22.12/21.80 for the joint model and 20.69/20.17 for the direct S2T approach, affirming the superiority of transformers.

Moreover, transformers have notably improved both G2T and S2G2T translations. Remarkable improvements in G2T, with BLEU-4 scores nearing 25.00, were reported by Yin &Read (2020) and Camgöz et al. (2020), indicating a substantial enhancement. In S2G2T, utilizing a sophisticated multistream recognition engine alongside a transformer, the authors of Yin &Read (2020) achieved state-of-the-art performance, with BLEU-4 scores of 22.47/24.00 for a single model and 24.68/25.40 for an ensemble, showcasing transformers as the definitive method for future SLT research.

4.3 A novel doubly stochastic SLT Transformer

In this section, we introduce a novel methodology for sign language translation that integrates the latest advancements in the field. Similar to earlier models, our approach is based on a Transformer architecture. However, in contrast to conventional deterministic Transformer networks, we propose a sophisticated model enriched by the integration of powerful stochastic components.

Our work aims to develop an SLT strategy that significantly enhances the accuracy of sign language translation. Our primary objective is to create an end-to-end SLT model that

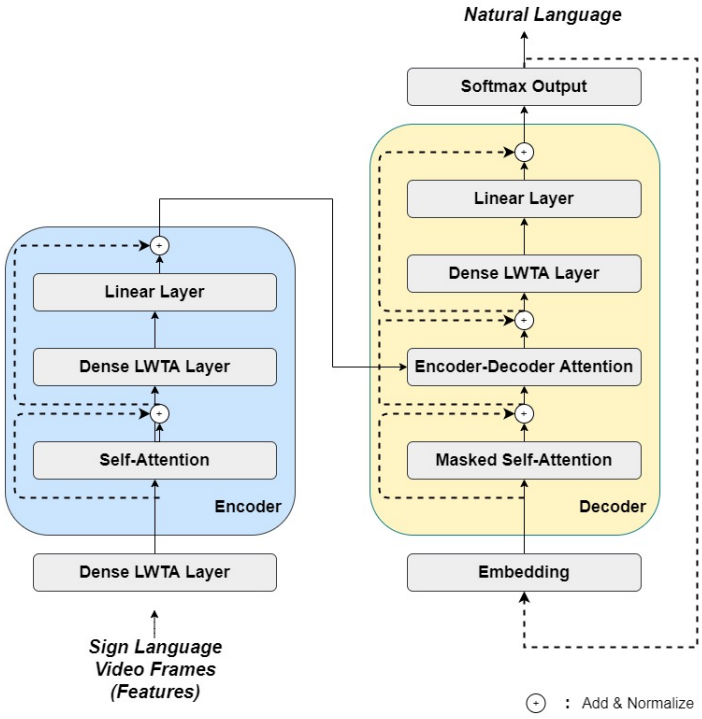

Figure 4.1: The Proposed Transformer network for end-to-end SLT.

reduces reliance on gloss annotations, thus avoiding their use as an intermediary recognition step in the S2G2T paradigm or as a concurrent task to optimize learned representations (S2(G+T)). Glosses are only used in an indirect fashion during the pretraining of the feature extraction engine. This approach is crucial for advancing the field, as generating gloss annotations for extensive datasets is exceedingly labor-intensive and costly. Furthermore, we aim to offer an SLT solution that requires less memory during inference, essential for the practical deployment of this technology.

To achieve this, we propose a Transformer network with a novel type of layers that combines: (i) local winner-takes-all (LWTA) layers with stochastic winner sampling, instead of conventional ReLU layers, (ii) stochastic weights with posterior distributions estimated via variational inference, and (iii) a weight compression technique at inference time that exploits estimated posterior variance to perform massive, almost lossless compression. We demonstrate that our method achieves state-of-the-art on the prominent SLT benchmark, PHOENIX 2014T, while significantly reducing the memory footprint compared to existing methods.

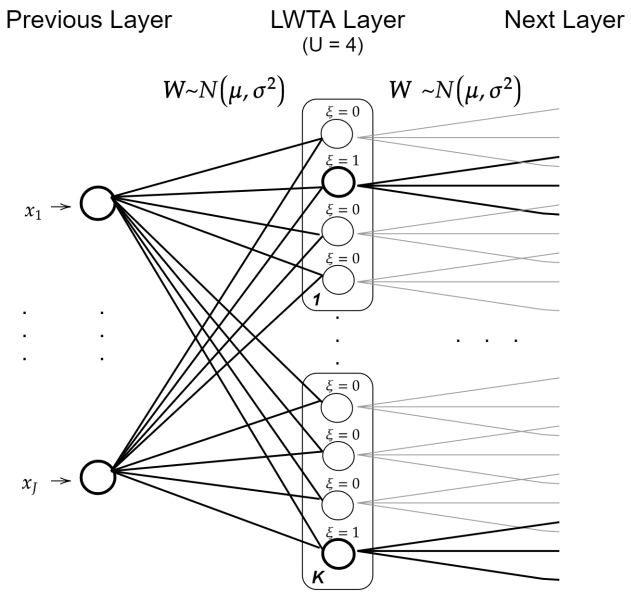

Figure 4.2: A graphical illustration of the proposed LWTA layers. Rectangles depict LWTA blocks, while circles therein represent competing linear units. The winner units are denoted with bold contours ($\xi = 1$). All edges correspond to Gaussian-distributed weights.

4.3.1 The proposed stochastic SL Transformer with linear competing units.

Key element of the proposed the model is the injection of the Stochastic "Local Winner Takes All" (LWTA) layer (Panousis et al., 2019a) in the transformer modality. An LWTA layer comprises linear units and introduces non-linear behavior by means of stochastic competition within blocks of layer neurons. Within a block of competitors, only one neuron, labeled as the "winner," is activated; winner selection is probabilistic. All other neurons remain inactive and pass zero values4.2.

In a more formal notation let us denote as $\boldsymbol{x} \in \mathbb{R}^J$ an input representation vector fed to some dense ReLU layer of a Transformer network, comprising J features. This layer is presented with a linear combination of the inputs, obtained via a weights matrix $\boldsymbol{W} \in \mathbb{R}^{J \times K}$, and produces an output vector $\boldsymbol{y} \in \mathbb{R}^K$, which is fed to the subsequent layer. In our approach, this mechanism is replaced by the introduction of LWTA blocks, each containing a set of competing linear units. The layer input is originally presented to each block, via different weights for each unit; thus, the weights of the connections are now

organized into a three-dimensional matrix $\boldsymbol{W} \in \mathbb{R}^{J \times K \times U}$, where K denotes the number of blocks and U is the number of competing units therein.

Under our approach, within each block these linear units compute their activations; for the uth unit in the kth block, we obtain the sum $\sum_{j=1}^{J}(w_{j,k,u}) \cdot x_j$. Then, the block selects one winner unit on the basis of a *competitive random sampling* procedure (described next), and sets the rest to zero. This way, we yield a *sparse* layer output, encoded into the vectors $\boldsymbol{y} \in \mathbb{R}^{K \cdot U}$ that are fed to the next layer.

In the following, we represent the outcome of local competition between the units in each block via the discrete latent vectors $\boldsymbol{\xi} \in \text{one\_hot(U)}^K$, where one_hot(U) is an one-hot vector with U components. These denote the winning unit out of the U competitors in each of the K blocks of a proposed layer, when presented with some input. Using this notation, the output reads

$$[\boldsymbol{y}]_{k,u} = [\boldsymbol{\xi}]_{k,u} \sum_{j=1}^{J}(w_{j,k,u}) \cdot x_j \ \in \mathbb{R} \tag{4.1}$$

where we denote as $[\boldsymbol{h}]_l$ the lth component of a vector $\boldsymbol{h}$. As we observe, at each time, only one (linear) unit in each LWTA block passes its output to the next layer, while the rest are zeroed out.

Let us now examine the statistical properties of the latent indicator vector $\boldsymbol{\xi}$. To enable data-driven competition between the units within an LWTA block, we postulate that the probability of a unit being sampled as the winner increases with the value of its (linear) output. In other words, we consider sampling from a Discrete posterior to select the winner at each time. On the basis of this rationale, we postulate that, a posteriori, it holds

$$q([\boldsymbol{\xi}]_k) = \text{Discrete}\Big([\boldsymbol{\xi}]_k \Big| \text{softmax}\big(\sum_{j=1}^{J} [w_{j,k,u}]_{u=1}^{U} \cdot x_j \big) \Big) \tag{4.2}$$

where $[w_{j,k,u}]_{u=1}^{U}$ denotes the vector concatenation of the set $\{w_{j,k,u}\}_{u=1}^{U}$.

On this basis, we obtain a novel variant of Transformer networks, the main operating principles of which are depicted in Fig. 4.3. We observe that the proposed network entails

statistical inference arguments, which bring to the fore stochastic activation principles. Drawing from this inspiration, we proceed to derive a full Bayesian treatment of the obtained network, by also considering that the network parameters themselves are governed by statistical principles. Specifically, we postulate that, throughout the network, all trainable weights are random variables; their (posterior) distributions can be estimated in data-driven fashion. For simplicity, we seek to derive (approximate) independent Gaussian posteriors over the set of trainable weights, $\boldsymbol{w}$:

$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)) \tag{4.3}$$

where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}^2$ is the variance of the Gaussians.

This concludes the formulation of the proposed Stochastic Transformer networks with competing linear units.

4.3.2 Training and inference algorithms

Overview

Our SLT model trains the entire Transformer network from scratch. The input modality consists of frame-wise feature sequences obtained from video frames. These frame-wise features are derived from a pretrained Inception network, in a manner similar to previous studies. This input is initially fed to an LWTA (Local Winner-Takes-All) layer (Panousis et al., 2019a), producing spatial embeddings that are then fed to the encoder of our proposed Transformer network, as illustrated in Fig. 4.7. The output modality, generated by the decoder part of our network, provides natural language interpretations. At each step, the decoder is presented with the previous word, initially processed by a standard linear embedding layer. The entire model is trained end-to-end, implementing LWTA blocks with 4 units each.

Training

To train the proposed model, we resort to maximization of the resulting evidence lower-bound (ELBO) of the model. To this end, we need to introduce appropriate prior assumptions regarding the distributions of the winner indicator latent variables, $\boldsymbol{\xi}$ on each LWTA layer, as well as the trainable weights, $\boldsymbol{w}$, throughout the network. For convenience, we postulate a priori spherical Gaussian weights of the form $p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and a symmetric Discrete prior over the winners: $[\boldsymbol{\xi}_n]_k \sim \text{Discrete}(1/U)$.

Introducing a mean-field (posterior independence) assumption across layers, we yield the following ELBO:

$$\mathcal{L}(\phi) = \mathbb{E}_{q(\cdot)}\big[CE\big] - \text{KL}\big[\ q(\{\boldsymbol{\xi}\})\ ||\ p(\{\boldsymbol{\xi}\})\ \big] - \text{KL}\big[\ q(\{\boldsymbol{w}\})\ ||\ p(\{\boldsymbol{w}\})\ \big] \tag{4.4}$$

where $\phi = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$ is the set of the means and variances of the Gaussian weight posteriors, trained throughout the network in an end-to-end fashion.

In this expression, $\mathbb{E}_{q(\cdot)}[CE]$ corresponds to the (negative) posterior expectation over the standard categorical cross-entropy error, used for training conventional Transformer networks. In a more analytical expression it reads as:

$$\mathbb{E}_{q(\cdot)}[CE] = -\mathbb{E}_{q(\cdot)}\big[\log p(\mathcal{D}|\{\boldsymbol{w}, \boldsymbol{\xi}\})\big] \tag{4.5}$$

On the other hand, KL stands for the Kullback-Leibler divergences pertaining to the model latent variables, i.e. the winner unit indicator latent variables, $\boldsymbol{\xi}$, and the connection weights, $\boldsymbol{w}$.

$$\text{KL}[q(\boldsymbol{\xi})||p(\boldsymbol{\xi})] = \sum_{\forall \boldsymbol{\xi}} \beta_\xi \sum_{i=1}^{U} q(\xi_i) \log\left(q(\xi_i)U\right) \tag{4.6}$$

$$\text{KL}[q(\boldsymbol{w})||p(\boldsymbol{w})] = \sum_{\forall \boldsymbol{w}} \beta_w [\frac{\boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2}{2} - \log \boldsymbol{\sigma} - \frac{1}{2}] \tag{4.7}$$

In these expressions, the factors β_ξ and β_w are heuristic hyperparameters needed to scale appropriately the KL contribution of each layer.

As we observe, the model ELBO entails posterior expectations over the network latent

variables, $\boldsymbol{\xi}$ and $\boldsymbol{w}$. These cannot be computed analytically. Therefore, all the posterior expectations in the ELBO are computed by drawing Monte-Carlo (MC) samples under (i) the standard reparameterization trick for the postulated Gaussian weights, $\boldsymbol{w}$; and (ii) the Gumbel-Softmax relaxation trick (Maddison et al., 2017; Jang et al., 2017) for the latent winner indicator variables of the LWTA layers, $\boldsymbol{\xi}$. On this basis, ELBO maximization is performed using standard off-the-shelf, stochastic gradient techniques; specifically, we adopt Adam (Kingma &Ba, 2014b).

Inference

Let us now turn to the inference algorithm of our network. At inference time, we *directly draw samples* from the trained posteriors of the *winner selection latent variables*, $\boldsymbol{\xi}$, of the LWTA layers, as well as the trained weight posteriors, $\boldsymbol{w}$, throughout the network. Thus, differently from previous work in the field, the proposed Transformer networks are characterized by a *doubly stochastic* nature, stemming from two different sampling processes. On the one hand, we implement a data-driven random sampling procedure (by sampling from $q(\boldsymbol{\xi})$) to determine the activations of Dense layers in the network (LWTA layers). In addition, we infer the weight values, throughout the network, again based on sampling from the trained posteriors $q(\boldsymbol{w})$.

Following standard practice, we approach inference in an autoregressive manner. We construct the sentence by predicting each word using the previous predictions as input. In detail, inference is performed by sampling the $q(\boldsymbol{\xi})$ and $q(\boldsymbol{w})$ posteriors a total of $S = 4$ times for each word prediction, and averaging the corresponding $S = 4$ sets of output logits (*Bayesian averaging*).

$$y_{m+1} = \frac{1}{4}\sum_{i=0}^{4} F(x, \{y_i\}_{i=0}^{m} | \{w, \xi\}); \;\; w \sim q(\boldsymbol{w}) \, , \xi \sim q(\boldsymbol{\xi}) \tag{4.8}$$

Furthermore we note that during validation, we perform a simple greedy search, while for the final test results, we employ beam search with optimized hyperparameters.

4.3.3 A compression scheme

According to the current standards (Sites, 2008), computers represent real numbers by a set of bits divided into 3 different subsets: a single sign bit, a set of eb exponent bits, and a set of pb significant precision bits. Then, the stored value is expressed as a product of three factors:

$$\text{value} = (-1)^{sign} \times 2^{exponent} \times (1 + mantissa) \tag{4.9}$$

The values of the three variables in this expression (sign, exponent, mantissa) are derived from the values of the corresponding bit subsets. The first variable, which defines the sign, is directly connected to the corresponding bit, giving positive values for sign = 0 and negative values for sign = 1. The exponent, the second factor, determines the range of values that can be represented. Finally, the third term, known as the mantissa, defines the floating-point precision. The exact formulations that connect these values with the corresponding bits are expressed as:

$$exponent = 2^{\sum_{i=1}^{eb} b_i \times 2^{i-1} - 2^{eb} - 1} \tag{4.10}$$

$$mantissa = \sum_{i=1}^{pb} b_{pb-i} \times 2^{-i} \tag{4.11}$$

where b_i denotes the ith bit in the corresponding subset. Typical machine learning implementations (e.g., PyTorch (Paszke et al., 2019)) employ 8 exponent bits and 23 precision bits (float32 format)

It is now well-established that a variational Bayesian treatment of deep network weights allows for significantly reducing the used bits without damaging accuracy (Panousis et al., 2019a; Louizos et al., 2017). Specifically, the obtained posterior variance of the network weights, σ^2, constitutes a measure of uncertainty in their sampled values. The higher the associated uncertainty the more the fluctuation of their values. One can leverage this uncertainty information to assess which precision bits, out of the pb available, are significant, and remove the ones which fluctuate too much under approximate posterior sampling. In addition, combining posterior mean, μ, and variance information allows for

estimating a confidence interval, that is an interval that sampled weight values may lie within with high probability. Using this information, we can also reduce the number of used exponent bits, eb. In our work, we perform both these reductions on a layer-wise basis. To this end, we consider the minimum posterior variance σ^2 of the weights within a layer, as well as the maximum μ values.

4.3.4 Feature Extraction

As previously discussed, the proposed sign language model processes input sign language videos on a frame-by-frame basis. Ideally, a sign language translation (SLT) model should be capable of undergoing the entire training cycle from scratch. This implies that the feature extractor, the part of the network responsible for extracting spatial information from video frames, should be trainable concurrently with the remainder of the model. However, this approach has been proved technically impossible, mainly for 2 technical reasons: (i) The memory requirements for training such architectures significantly exceed the capabilities of a standard GPU. (ii) Even with the potential use of multiple GPUs and/or particularly advanced hardware, the complexity of the task combined with a lack of sufficient data leads to suboptimal results.

To address these challenges, we opted for a feature extraction engine utilizing pre-trained networks. Initial experiments with generic pre-trained networks, such as EfficientNets and Inception networks pre-trained on ImageNet, yielded low-value results. To enhance performance, we utilized specialized pre-trained networks on tasks related to sign language. Specifically, we followed the setup proposed in (Koller et al., 2019), pre-training a model combining a CNN (Inception)-LSTM architecture for sign language recognition along with mouth and hand shape detection. From this model, we only utilized the spatial component (i.e., the Inception Network) as our pre-trained feature extractor, which resulted in significantly improved outcomes.

4.3.5 Experimental Results

In this Section, we perform the comparative assessment of our approach. To this end, we use PHOENIX 2014T dataset (Camgoz et al., 2018); this constitutes the most used benchmark in the recent literature. Hence, this benchmark selection allows for optimal and full comparability of our results with the recent related work in the field. As mentioned before the used dataset contains German SL videos of weather forecasts, and corresponding translations into the German spoken language. They are obtained from 9 different speakers.

4.3.5.1 Experimental setup

The whole resulting model is trained in an end-to-end fashion, as described previously. We implement our method considering LWTA blocks of $U = 4$ units each. All trained Transformers use embedding sizes of 512 and 8 attention heads. Weight posterior means and variances are initialized by means of Kaiming uniform initialization (He et al., 2015). Conventional models, for which we obtain point-estimates (as opposed to weight posteriors), are initialized by employing Xavier normal (Glorot &Bengio, 2010). Gumbel-Softmax temperature is set to $T = 1.69$ for training and $T = 0.01$ for inference. In all cases, we use Adam (Kingma &Ba, 2014b) with a learning rate of 0.001 ($\beta_1 = 0.9, \beta_2 = 0.998$), and a batch size of 32. During training, we evaluate the networks on the validation set every 80 iterations, and decrease the learning rate by 20% if the validation does not improve for 5 consecutive iterations. Training ends when the learning rate falls bellow a minimum of 0.0001. This evaluation process during network training is performed via greedy decoding. At inference time, evaluation on the test set is performed by means of Beam-Search; we perform several runs to determine optimal beam-size in all cases. Our main reference metric for assessing translation quality is the BLEU-4 score.

Our implementation is developed in Pytorch (Paszke et al., 2019), and based on the "JOEYNMT" (Kreutzer et al., 2019), "Sign language transformer" (Camgöz et al., 2020), "Bayesian Compression for Deep Learning" (Louizos et al., 2017), and "nonparametric Bayesian local-winner-takes-all" (Panousis et al., 2019a) frameworks.

Before discussing our results, we first present some of the latest state-of-the-art method-

Table 4.1: Previous State-of-the-art BLEU-4 scores

Model	Dev	Test
S2T (Camgöz et al., 2020)	20.69	20.17
S2(G+T) (Camgöz et al., 2020)	22.12	21.80
G2T (Camgöz et al., 2020)	25.35	24.54
S2G2T-STMC (Yin &Read, 2020)	22.47	24.00
S2G2T-STMC ensemble (Yin &Read, 2020)	24.68	25.40

ologies on the considered benchmark, published before the release of our method, for further reference. Table 4.1 summarises the BLEU-4 scores of those models. The first state-of-the-art model we consider in our experimental evaluations is the sign-to-text transformer (S2T) (Camgöz et al., 2020). Our SLT method presented in this paper largely extends upon this method; thus, we consider this approach as our Baseline.

In addition, we consider three further Transformer-based models, namely a gloss-to-text (G2T) (Camgöz et al., 2020), a sign-to-gloss-and-text (S2(G+T)) (Camgöz et al., 2020), and a sign-to-gloss-to-text (S2G2T) (Yin &Read, 2020) model. These methods obtain higher BLEU scores than the basic S2T; the last one actually yields the highest performance reported to-date in the considered benchmark. However, as mentioned before, these networks require the possible gloss sequences which may be hard to obtain for large training datasets. Specifically, S2(G+T) takes advantage of glosses as a parallel task that facilitates the encoder to obtain better representations; S2G2T utilizes them as an intermediate step, while G2T uses gloss input to obtain natural language (this renders it the least relevant to a real-world SLT task, as it assumes availability of a system that allows for perfect gloss recognition). Additionally, we emphasize that S2G2T employs the computationally burdensome STMC 3-channel recognition network (Zhou et al., 2020), while we process the whole frame as a single channel.

4.3.5.2 Main results

In Table 4.2, we summarize the performance of our model for network configurations of varying depth. In our setup, an encoder (decoder) of depth H means a module comprising H consecutive submodules of the form depicted on the left (right) hand side of Fig. 4.3(a).

Table 4.2: Proposed Approach: BLEU scores for varying depths.

Set	Encoder	Decoder	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Dev	1	1	48.67	35.34	27.3	22.03
	2	2	49.12	36.29	28.34	**23.23**
	3	3	45.68	32.87	25.72	21.66
Test	1	1	47.47	34.75	26.8	21.85
	2	2	48.61	35.97	28.37	**23.65**
	3	3	45.84	33.40	25.72	21.29

Table 4.3: Network compression as per Section 3.4: effect on memory requirements and translation quality.

Depth encoder-decoder	Average Bits	Memory Reduction	Dev BLEU-4	Dev change	Test BLEU-4	Test change
1 - 1	9.4	70.6%	21.66	-1.6%	22.05	+0.9%
2 - 2	8.8	72.3%	23.09	-0.6%	23.52	-0.5%
3 - 3	8.7	73.0%	20.82	-3.8%	20.77	-2.4%

Comparing the best performance reported therein with the summary of state-of-the-art results in Table 4.1, we observe that our method outperforms the corresponding S2T baseline approach by 3.48 BLEU-4 scores on the test set. In this context, the best configuration under the proposed modeling approach seems to be the (2-2); this achieves BLEU-4 scores as high as 23.65 on the test set. This performance is superior to the S2(G+T) hybrid network as well, which yields 21.80 BLEU-4 on the test set. This outcome becomes even more prominent if we consider that S2(G+T) imposes much higher computational burden, and most importantly, it requires the possible sequences of glosses as groundtruth.

Subsequently, we examine network compression. By employing the layerwise compression scheme outlined in Section 3.4, we manage to reduce the average required bits for storing network parameters from 32 to less than 10. This fact implies a memory usage of around 30% that of the baseline SLT Transformer network of (Camgöz et al., 2020). In Table 4.3, we present the average required number of bits throughout the layers of our network. In addition, we show how the compressed network performs in terms of the obtained BLEU-4 scores. These scores are obtained by compressing network parameters, and then re-running inference. Our results show that our compressed network incurs a

negligible trade-off in translation accuracy, for massively lower memory needs.

Finally, we turn to the S2G2T ensemble (Yin &Read, 2020), which still performs better than our approach, yielding a BLEU-4 score of 25.40 (c.f., Table 4.1). The key element that renders S2G2T ensembles so potent is the utilization of ensemble decoding. This consists in averaging the predictions of different networks, in order improve the eventual translation quality. Thus, it is worth to examine whether an ensembling scheme can also improve the BLEU-4 scores of our method. To this end, we repeat our experiments with the (2-2)-version, training 16 different network instances with different random seeds. We use the best performing $L = 4$, $L = 8$, $L = 12$ as well as all of the so-obtained networks ($L = 16$) to perform ensemble-decoding.

Table 4.4: BLEU-4 scores with Ensemble-Decoding.

L	32 bit		Reduced	
	Dev	Test	Dev	Test
4	24.02	24.84	24.23	24.52
8	24.88	25.59	24.52	25.33
12	24.96	25.71	24.55	25.76
16	24.88	**25.80**	24.73	25.68

In Table 4.4, we present the obtained BLEU-4 scores. As it was expected the ensemble model scores are superior to the single model approach and results are getting better while increasing the ensemble size. With $L = 16$, our approach yields a BLEU-4 score of 25.80; this was the best BLEU-4 score reported in the literature on the considered dataset until late 2022. We emphasize that we obtain this performance without making use of any predefined gloss sequences that need alignment in the Transformer network pipeline, contrary to (Yin &Read, 2020). Then, we repeat our ensemble-decoding experiment using the technique of Section 3.4 to perform parameter compression. We obtain a memory footprint reduction similar to the second line of Table 4.3. As we show in Table 4.4, for a memory footprint reduced by approximately 70%, our method remains competitive.

4.3.6 Ablation study

In this section we analyse in greater detail key elements of the proposed model.

4.3.6.1 Determining the Optimal Block Size U

The size of the competition blocks in LWTA layers is considered a significant and potentially impactful hyperparameter. As previously mentioned, in our main experimental investigation, we employed a block size of $U = 4$, which has been identified as optimal in our context, diverging from earlier works such as (Panousis et al., 2019a, 2021b,a). To substantiate this selection, we have rigorously reassessed our proposed model across various values of U, specifically for $U \in [2, 4, 8, 16, 32]$. Given that the $(2-2)$ version of our network demonstrated the highest precision, our subsequent experiments focus on this particular configuration. The results of this study, reported in Table 4.5, confirm our previous statement.

Table 4.5: Block Size (U) comparison (BLEU-4 scores).

Block Size	32 bit		Reduced	
	Dev	Test	Dev	Test
$U = 2$	22.99	22.82	23.12	22.37
$U = 4$	23.23	**23.65**	23.09	23.52
$U = 8$	22.28	22.96	22.35	22.72
$U = 16$	22.32	22.52	22.00	22.34
$U = 32$	21.55	21.16	21.43	20.81

4.3.6.2 How standard non-linear activation would perform?

A naturally coming question is what the model's performance would be if we replace the LWTA layers with popular approaches that introduce non-linearity through ReLU based layers or other popular deterministic activation functions. To clarify these questions we now scrutinize the proposed stochastic competition-based activation functions. Specifically, we re-implement our method using ReLU and other popular activation functions in place of the proposed LWTA layers. It is important to note that, despite this alteration, we continue to adopt a full variational Bayesian approach, inferring Gaussian weight posteriors.

The results in Table 4.6 demonstrate the experimental outcomes. The classical ReLU activation emerges as slightly superior, achieving a BLEU-4 score of 22.61, with other variants exhibiting comparable performance. This finding underscores that our LWTA

Table 4.6: Activation function comparison (BLEU-4 scores).

Activation	32 bit		Reduced	
	Dev	Test	Dev	Test
ReLU	22.42	22.61	22.17	22.67
Elu	22.63	22.56	22.19	22.32
SiLU	22.73	22.33	22.23	21.99
GeLU	22.45	22.49	22.17	22.36
Swish	22.01	21.03	21.84	21.24

activation, especially with four units per block, offers the highest overall performance, surpassing widely used ReLU and other conventional activation functions by over 1 BLEU-4 unit (with $U = 4$). Furthermore, a combined analysis of Tables 4.6 and 4.5 reveals that LWTA outperforms ReLU and other studied activation functions for both $U = 2$ and $U = 8$ reinforcing the superiority of our proposed method.

4.3.6.3 Does the variational Bayesian treatment of network weights contribute to SLT accuracy?

Conversely to the experiments of the previous Section, it is also important to examine whether training full variational posteriors over the network weights does actually offer tangible gains in terms of translation accuracy. To this end, we re-implement our method, making full utilization of the proposed stochastic LWTA activations, but obtaining conventional point-estimates over the network weights. Thus, the set of network weights, $\boldsymbol{w}$, becomes now a parameters set that we optimize during training. Specifically, network training now reduces to maximization of the following ELBO expression:

$$\mathcal{L}(\boldsymbol{w}) = \mathbb{E}_{q(\boldsymbol{\xi})}\big[\log p(\mathcal{D}|\{\boldsymbol{\xi}\})\big] - \mathrm{KL}\big[\, q(\{\boldsymbol{\xi}\}) \,||\, p(\{\boldsymbol{\xi}\}) \,\big] \tag{4.12}$$

In Table 4.7, we provide our results, again considering the (2-2)-version of our method, which constitutes its best-performing configuration. Our findings show that, even with point-estimates, we manage to score 2 BLEU-4 units above the S2T Baseline. This outcome is clearly inferior to our full-fledged model. Therefore, we deduce that the variational Bayesian treatment of connection weights, throughout the proposed network,

offers important SLT accuracy gains. This comes in addition to allowing for massive memory savings, by following the rationale of Subection 3.4.

Table 4.7: Comparison of variational Gaussian weights to point-estimates (BLEU-4 scores).

Weights Type	32 bit		Reduced	
	Dev	Test	Dev	Test
Point-Estimates	22.54	22.34	-	-
Variational Gaussian	23.23	23.65	23.09	23.52

4.3.6.4 Effect of sample size S at inference time

As we explained before, inference is performed through Bayesian averaging with a sample size of $S = 4$. It is useful to examine how this selection affects BLEU-4 scores. Thus, we repeat our experiments with the (2-2) configuration for different S values, and report the outcomes in Table 4.8. In addition, we exploit this opportunity to examine how the model behaves if we do not sample winners and connection weights; instead we pick the unit the yields the maximum posterior probability $q(\boldsymbol{\xi})$, and use the connection weight means, $\boldsymbol{\mu}$, to perform feedforward computations ("Deterministic" scenario).

Table 4.8: Sample size effect (BLEU-4 scores).

Sample Size	32 bit		Reduced	
	Dev	Test	Dev	Test
Deterministic (1)	22.99	22.83	22.21	22.31
1	23.45	23.05	22.66	23.03
4	23.23	23.64	23.09	23.52
16	23.34	**23.95**	23.14	23.39

We observe that the increased sample size indeed improves model stability and performance. More specifically, a large sample size $S = 16$ can slightly increase BLEU-4, while $S = 1$ produces a much lower performance than the proposed $S = 4$ size. However, large sample size also increases the model's inference time, and it may be impractical, especially for $S >> 16$. Moreover, we can see that a deterministic version of our model is clearly inferior to stochastic operation, even if we draw just one sample, $S = 1$.

4.3.6.5 Computational Complexity

In table 4.9, we compare the computation times of our approach with relevant fully deterministic method.

Table 4.9: Computation Time on a single Quadro P5000 16GB.

	Deterministic	Stochastic
Training (per batch)	0.2s	0.5s
Inference (single video)	0.04s	0.05s

The differences in training time are due to the increased number of trainable parameters, namely the trainable variances of the Gaussian weights, and *not* due to Bayes. We emphasize that these trainable variances are used for proper weight compression, and are eventually discarded after training. Crucially, *convergence took almost the same number of epochs* with the baseline. Figure 4.3 illustrates both the learning curves and proves the latter statement and the smooth convergence of our network. The computational efficiency of our approach becomes even more apparent when we check inference times, which are comparable with the baseline Transformer when using $S = 4$ samples (that's the best trade-off between accuracy and complexity).

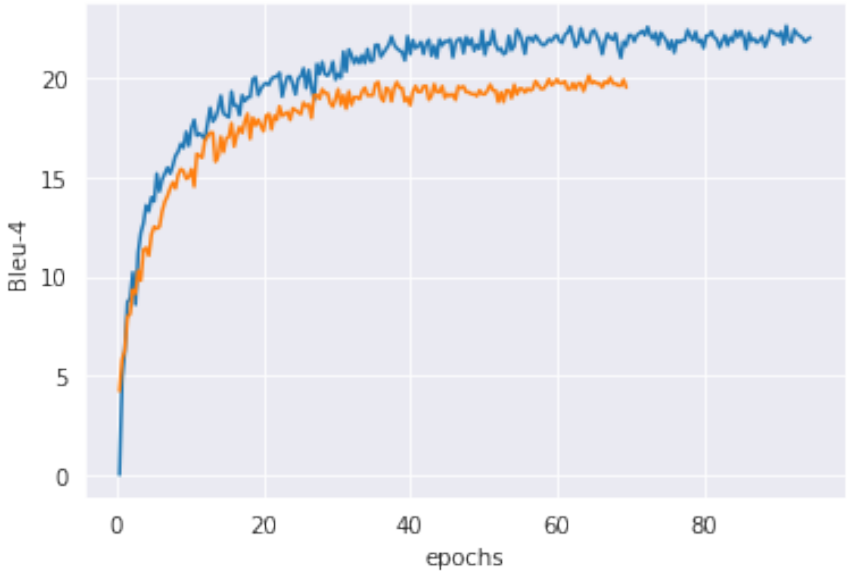

Figure 4.3: Convergence curves. Blue: proposed, Orange: baseline.

4.3.7 Qualitative investigation

From a qualitative perspective, besides the complexity of the task, our translations seem to be of acceptable quality (Table 4.10). There is a small number of syntactic and grammar errors; most of them are about locations and dates. Moreover, while in many cases the predicted sentence is syntactically different from the groundtruth, the resulting meaning remains similar.

Table 4.10: Translation examples including both original (German) followed by English translation: Reference (R), single model (S), and ensemble (E).

#	**Translation references and predictions**
1	**R:** im süden schwacher wind *in the south weak wind* **S:** der wind weht meist nur schwach *the wind is blowing mostly low* **E:** der wind weht im süden schwach bis mäßig *the wind blows weak to moderate in the south*
2	**R:** am freitag insgesamt viele wolken die regen bringen *on friday there were a lot of clouds that bring rain* **S:** am donnerstag viele wolken hier und da schauer *look at a lot of clouds here and there on thursday* **E:** am freitag gibt es viele wolken und gebietsweise schauer *on friday there are lots of clouds and showers in some areas*
3	**R:** ganz ähnliche temperaturen wie heute zwischen sechs und elf grad *very similar temperatures as today between six and eleven degrees* **S:** am bodensee heute nacht nur sechs bis elf grad *only six to eleven degrees on Lake Constance tonight* **E:** ähnliches wetter heute nacht *similar weather tonight*
4	**R:** im westen und nordwesten fallen einzelne schauer . *individual showers fall in the west and northwest* **S:** im westen und nordwesten gibt es im westen hier und da schauer . *in the west and northwest there are showers here and there in the west* **E:** im westen und nordwesten gibt es im westen einige schauer . *in the west and northwest there are some showers in the west*
5	**R:** in der neuen woche kühlt es dann bei wechselhaftem wetter deutlich ab . *in the new week it cools down considerably with changeable weather.* **S:** in der neuen woche wechselhaft und deutlich kühler . *in the new week changeable and clearly cooler.* **E:** in der neuen woche unbeständig und noch kühler . *in the new week unstable and even colder.*

4.4 Achieving a milestone in end-to-end Greek Sign Language Translation

Our model has demonstrated exceptional predictive capabilities and achieved state-of-the-art results on standard benchmarks within the field. However, its practical application value remains limited, a limitation shared by the majority of existing sign language translation engines. This constraint primarily arises from two issues related to the available datasets. Firstly, the nature of the Phoenix24T dataset, among other popular datasets, is highly specialized, focusing narrowly on weather forecasting. This specialization results in a restricted vocabulary and frequent repetition of phrases, which, while simplifying the development of SLT engines and facilitating translations closely matching the ground truth, also limits the engines' applicability to a broader range of topics. Secondly, although our method does not directly rely on gloss annotation for training, it necessitates such auxiliary data for pretraining the feature extractor. Given that gloss annotation is both costly and challenging to obtain for larger datasets, this represents a significant obstacle.

To address these challenges and advance towards a more comprehensive SLT solution, two key developments are necessary: (i) the creation of models that are entirely independent of gloss annotations, and (ii) the acquisition of more diverse datasets featuring a broader range of content and more realistic vocabulary distribution. In response to these needs, we have modified our stochastic sign language transformer to operate without reliance on gloss annotations at any level. Fortunately, this is possible through a reconsideration of the spatial feature extraction technique combined with appropriate hyperparameter tuning. Furthermore, recognizing the scarcity of realistic datasets in existing literature, we introduce a novel dataset, the Greek Elementary School dataset. The creation of a new SLT dataset is a particularly long and complex procedure, requiring significant effort from a team with a wide range of expertise. This attempt was compiled with the invaluable support of the Hellenic Federation of the Deaf, marking a significant step forward in enriching the resources available for SLT research (Voskou et al., 2023).

Table 4.11: Elementary23 statistics vs Phoenix2014T.

	Phoenix2014T	Elementary23
Signers	9	9
Total Hours	25	71
Total Frames	$\approx$ 1.1M	$\approx$ 6.3M
Sentences	8257	29653
Vocabulary	2887	23204
Singletons	1077	10126
Resolution	210×260	1280×720
FPS	25	25

4.4.1 Elementary23 Dataset

4.4.1.1 Core Elements

The introduced Greek Elementary School dataset [1], dubbed Elementary23, constitutes a noteworthy contribution to the existing body of literature due to its exceptional quality and the substantial number of examples included. Table 4.11 presents a comparative analysis of Elementary23 and the widely utilized Phoenix2014T dataset. As we show, Elementary23 exhibits superior technical and linguistic characteristics.

4.4.1.2 Collection Procedure

The Elementary23 dataset stands in contrast to many relevant datasets as it was not built by annotating preexisting sign language (SL) videos from online or other sources. Instead, it was assembled through the recording of sign language interpretations of authentic elementary school content. Furthermore the final content was rigorously curated and selected in cooperation with SL experts, ensuring its high impact and practical value to the deaf community and students.

The recordings for the Elementary23 dataset were made in an environment optimally suited for the task. This included a fixed single-color background and ideal lighting conditions (see Fig. 4.4). Professional-grade cameras and equipment were used to record videos at 720p resolution and a frame rate of 25fps.

[1] https://zenodo.org/record/7847052

Figure 4.4: Example of Elementary23 Sign Language Video Frames

The material was organised into sentences/phrases and then assigned to nine signers, all proficient users of Greek Sign Language with extensive knowledge and experience. As a result, the dataset is of exceptional quality, with minimum kinesthesiological and technical errors. The distribution per signer is illustrated in Figure 4.5.

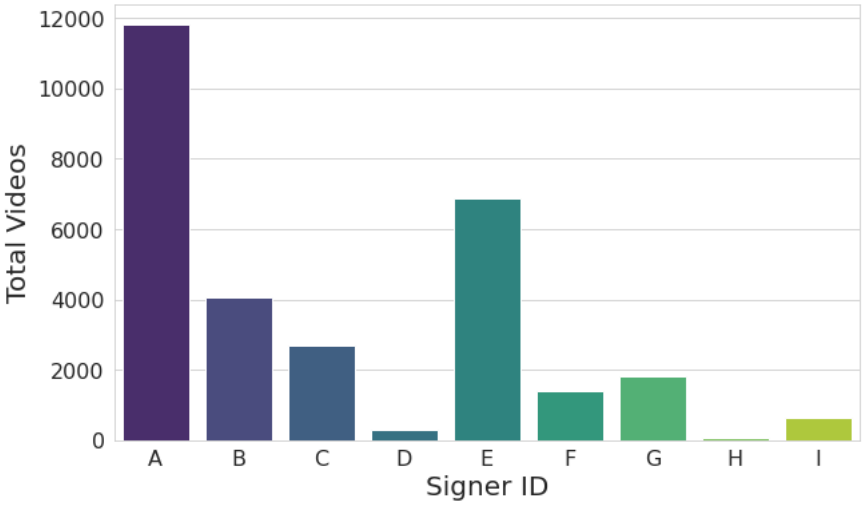

Figure 4.5: Distribution of video-translation pairs per signer

4.4.1.3 Content and Vocabulary

A Notable aspect of Elementary23 is its broad thematic spectrum. As previously mentioned the dataset is based on the official syllabus of Greek elementary schools, including the subjects of Greek Language, Mathematics, Religion Study, Environmental Study, History, and Anthology. The combination of those subjects ensures a sizeable lexicon of 23,204 words; yet, it is important to note that each individual subject contributes broad content

Table 4.12: Number of examples per Subject and vocabulary metrics

	Vocabulary	Examples
Anthology	14741	4158
Greek Language	14345	9499
Mathematics	6457	6583
History	7716	2067
Envir. Study	9489	5521
Relig. Study	8087	1825

Table 4.13: Elementary23-SLT vs key Benchmarks

	Elementary23-SLT	Phoenix2014T	SWISSTXT-NEWS	VRT-NEWS
Sentences	8372	8257	6031	7174
Total Words	83327	99081	72892	79833
Vocabulary	8202	2887	10561	6875
Mean Freq.	10.16	34.3	6.9	11.6
Singletons	3327 (41%)	1077 (37%)	5969 (57%)	3405 (50%)
Rare words	6155 (75%)	1758 (60%)	8779 (83%)	5334 (78%)

and an extensive vocabulary. The statistics for each subject are presented in Table 4.12. The largest in terms of video-translation pairs quantity is the subject of the "Greek Language", which contains 9499 examples; the smallest one is "Religion Study", with 1,825 entries. Vocabulary-wise, the most comprehensive subject is Anthology which includes a total of 14,741 different words; on the other end, "Mathematics" have the smallest vocabulary with 6,457 words.

4.4.1.4 SLT Subset

The main motivation of Elementary23 is to contribute one of the largest Sign Language datasets paying particular emphasis to technical and linguistic excellence. However, the dataset is not necessarily an ideal candidate for training end-to-end SLT deep networks, at least not in its raw form. Specifically, there are aspects of Elementary23 that present significant modelling challenges for deep networks. These include the dataset's high word sparsity; the high number of singletons (words appearing only once in the corpus); the inclusion of particularly small phrases; and the limited number of frequently-appearing words. To be fair, this is not a problem with the dataset itself but rather a limitation of

modern deep networks, which typically require multiple examples to learn from data. To overcome this issue, in this work we also present an appropriate representative subsample of the dataset, which we dub Elementary23-SLT. This is more suitable for training end-to-end SLT deep networks, and has a size similar to Phoenix2014T and other recently published benchmark datasets.

The selection process was guided by three primary principles: (i) decrease the absolute number of singletons; (ii) increase the density of frequent words and bigrams; and (iii) keep content diverse. To achieve these targets, we first went through preliminary cleaning, whereby we eliminated singleton-only sentences. Afterwards, we ran a simple dynamic multi-round elimination process: At each round, we listed all sentences containing singletons but no frequent elements, and we eliminated approximately a quarter of them. We then recalculated word frequencies and redefined singletons and frequent words based on the surviving subset. At the end of each round, we confirmed that all six subjects were represented with a sufficient number of remaining examples, at least 10% of the original. We repeated this procedure several times, until we ended up with a sample size comparable to the standard benchmarks.

The aforementioned process left us with a sample of 7168 sentences. Subsequently, we split the data into the typical train, validation and test subsets. During the selection of the validation and test sets, we tried to avoid sentences shorter than four words long, and made sure to exclude all the sentences appearing twice by the same speaker. As an extra processing step, we augmented the training set with an additional 1,204 non-singleton single-word videos. Finally, for comparison reasons, we additionally performed a train-validation-test split on the entire dataset following similar principles regarding duplicated entries. We will be referring to this split as Elementary23-Raw. In both cases, all speakers may participate in all the subsets.

The final SLT set contains 8372 video-sentence pairs, organized as 7348 pairs for the training set, 512 for the validation set, and 512 for the test set. Through this operation, we produced an effective subset in the typical size spectrum, that retains the desired qualitative elements of the complete collection while exhibiting some improved quantitative

factors. Key statistics regarding the subset and benchmarks are stated in Table 4.13 and will be analysed later in this Section.

To enable the examination and exploitation of the data by the research community, we have ensured that Elementary23-SLT complies with established standards regarding size and structure. Additionally, the data will be available in a file format that is practical and consistent with prior works (Camgöz et al., 2020; Voskou et al., 2021). Specifically, we have created a JSON file comprising a list of dictionaries, each corresponding to a particular video-sentence pair. These dictionaries encompass all auxiliary elements, such as numbering and signer ID, as well as the principal input-output data; the latter consists of the frame-wise feature sequence and the corresponding translation in modern Greek.

4.4.2 Lexical Statistics and Benchmarks

While Phoenix2014T has gained popularity as a standard benchmark for SLT due to its ease of modelling, its appropriateness as a benchmark for comparing to the newly introduced dataset remains questionable. This is due to the following facts: i) it covers only a single topic, in contrast to the multi-subject nature of Elementary23; ii) vocabulary coverage is very limited; iii) it includes many sentences of similar structure and content, due to the weather forecasting's strict format; and iv) it includes auxiliary Gloss annotation.

To address these challenges, we are supplementing our benchmark schema with two more datasets: SWISSTXT-NEWS and VRT-NEWS. Containing 6031 and 7174 videos/sentences respectively, they're based on HD TV news videos. We chose them due to their similar size to Phoenix2014T and Elementary23-SLT, good video quality and their basis in European sign languages, ensuring directly comparable grammatical structures.

In Table 4.13, we present a detailed comparison of Elementary23-SLT and the three selected benchmark datasets on the grounds of various vocabulary-oriented metrics. These metrics were critical in determining the suitability of SWISSTXT-NEWS and VRT-NEWS as the primary benchmarks, since their statistics closely resemble those of Elementary23-SLT. Specifically, both SWISSTXT-NEWS and VRT-NEWS contain similarly sized vocabularies, with 10561 and 6875 sentences, respectively. Thus, they deviate no more than 25

% from our proposed subset of 8202 sentences. Furthermore, the percentage of rare words, defined as words that appear less than five times, ranges from 75 % to 83% for all three datasets. Finally, the mean word frequencies are comparable across these datasets, with the words in SWISSTXT-NEWS appearing an average of 11.6 times, 6.9 for VRT-NEWS, and 10.1 for Elementary23-SLT.

Notably, Phoenix2014T has a vastly reduced vocabulary size compared to the rest of the considered datasets, which constitutes a central constraint. Additionally, Phoenix2014T is characterized by a considerably higher mean word frequency equal to 34.3. This number is rooted in the limited vocabulary and allows for easier training since it narrows down verbal varieties. Furthermore, Phoenix2014T has the lowest percentage of rare words and singletons. In terms of singletons, Elementary23-SLT is a close second, with an increase of only 4%.

4.4.3 Translation Methology

For the Greek Sign Language Translation model, we utilize both the architecture we have previously proposed, namely the Stochastic LWTA Sign Language Transformer, and a corresponding transformer that employs fully deterministic methods for comparison purposes. In the following, we will be referring to our Transformer-based SLT model variant as the stochastic model. On the other hand, the deterministic variant will be dubbed simply as the deterministic model.

Given our current focus on real-life applicability over high rankings on standard benchmarks, our goal is to develop a fully gloss-free network. This objective necessitates a reevaluation of our approach to extracting spatial information and, consequently, an adaptation of the network's hyperparameters. We discuss these adjustments in detail in the subsequent section.

4.4.3.1 Landmarks and Trajectories

State-of-the-art SLT networks, including ours, often utilize convolutional subparts as feature extractors; these are pre-trained on sign language recognition datasets. Such

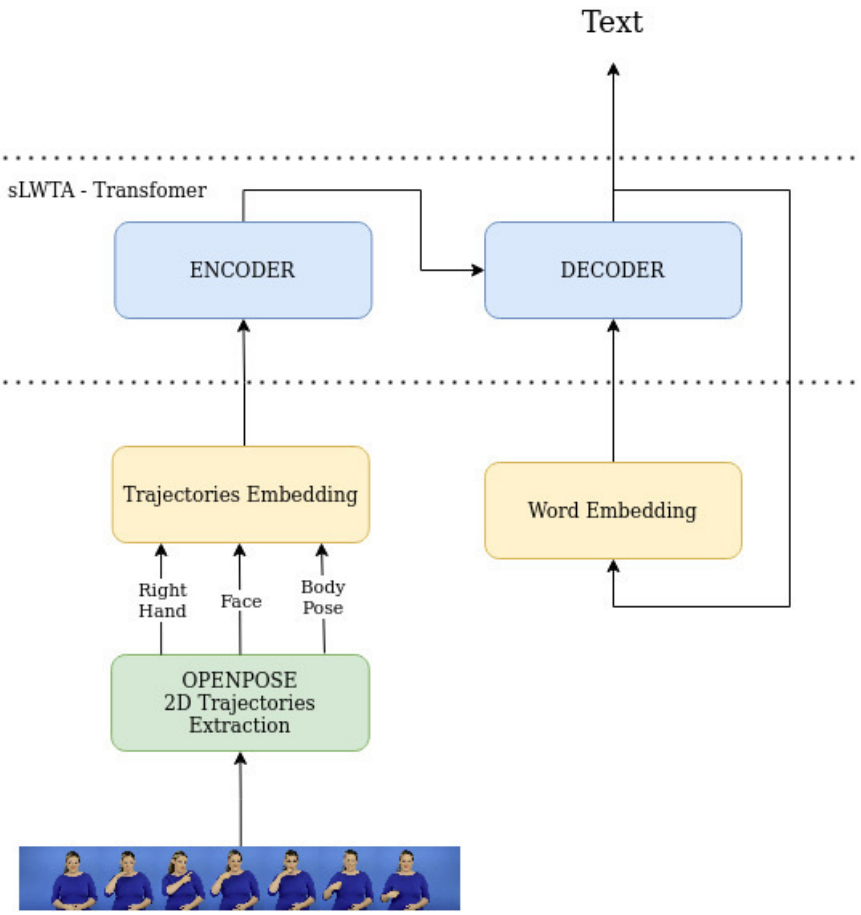

Figure 4.6: The suggested Sign to Text Transformer Network

subparts can extract spatial information from video frames by leveraging their prior knowledge of core sign language elements. However, this preprocessing phase requires laborious dataset annotation in terms of auxiliary Glosses. Therefore, this process is of reduced applicability: developed models can only generalize on other datasets of similar Gloss structure. In addition, it is incompatible with the Greek Elementary School dataset: its immense size renders the provision of Glosses completely out of scope.

To address this issue, we follow an alternative, yet established approach that involves using the OpenPose engine (Cao et al., 2017). The OpenPose engine is a convolutional neural network that has been trained to track and extract the trajectories of key human body parts. We use OpenPose to track landmarks related to the 2D positioning of upper body movements, facial expressions, and hand shapes; one can use these as input to a developed end-to-end SLT model. We further scrutinize the extracted features, by excluding components that appear not to contribute enough or exhibit persistently low

Figure 4.7: Body Landmarks - Trajectories

volatility, such as lower body landmarks. Figure 4.7 illustrates a representative example of this approach. The resulting vector effectively summarizes the video frames and serves as a sufficient source of information for subsequent network layers 4.6.

4.4.3.2 Training and Inference

For both training and inference algorithms are very similar to the slt model we descriped in the previous section. The training objective of the deterministic SLT model will be to minimise the cross-entropy error between each predicted word and the corresponding label, under a standard seq-to-seq rationale. When it comes to inference using the trained model, we again follow the usual practice and run autoregressive decoding, where words of each produced sentence are predicted one by one, and the decoder is presented the encoded representations and the previous predictions. Additionally, we use the beam search algorithm, the parameters of which are optimised on the validation set. Training of the stochastic variant is slightly more complex following the procedure we have described at the previous section.

Table 4.14: Results using deterministic and stochastic Transformers for both the entire dataset and SLT subset

Data Set	Dev/Test	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Raw	Dev	Stochastic	10.4	2.14	0.95	0.33
Raw	Dev	Deterministic	6.23	1.23	0.54	0.36
Raw	Test	Stochastic	11.50	2.85	1.05	0
Raw	Test	Deterministic	7.68	1.83	0.5	0
SLT	Dev	Stochastic	21.30	12.26	8.74	**6.67**
SLT	Dev	Deterministic	18.79	9.69	6.68	5.08
SLT	Test	Stochastic	19.99	11.10	7.68	5.69
SLT	Test	Deterministic	17.37	8.50	5.35	3.85

4.4.4 Experimental Results

4.4.4.1 Experimental setup

This section presents our experimental results, primarily focusing on the LWTA-Transformer's application on the SLT subset of the Elementary dataset. The suggested LWTA-Transformer version is a two-layer architecture with an embedding size of 256 and U=2.

Following the recommendations of (Camgöz et al., 2020), and (Paszke et al., 2019) we initialised the trainable parameters using Kaiming uniform for stochastic models (He et al., 2015), and Xavier normal for deterministic models (Glorot &Bengio, 2010). The Gumbel-Softmax temperature was set following the related theory in (Jang et al., 2017); we use a high temperature of $T = 1.00$ during training and a low $T = 0.01$ during inference. The rest of the training hyperparameters were either chosen based on the exact suggestions from the original papers of the models or optimised during our experimental investigation. The BLEU-4 score was used as the main evaluation metric to assess the quality of sign-to-text translation. Core parts of the model's implementation are modified versions of (Kreutzer et al., 2019; Camgöz et al., 2020; Voskou et al., 2021).

4.4.4.2 Quantitative Results

The central inquiries targeted in our experiments are (i) the identification of the network architecture that produces the best outcomes and prove the superiority of our stochastic LWTA-Transformer method; and (ii) quantifying the impact of our decision to use an SLT-suitable subset rather than working with the entire Elementary23 dataset. Table 4.14 presents the best results that were achieved for all the subcases.

Table 4.14 clearly demonstrates that using the Raw dataset resulted in very low BLEU scores for both the deterministic and stochastic models. Conversely, the usage of the SLT-subset resulted in significantly improved outcomes. The exclusion of problematic or unsuitable sentences and an appropriate data split apparently play a crucial role, making training less noisy and more focused on effective examples.

In terms of architecture, we compare two of the best approaches available, that is, the original S2T deterministic Transformer and its stochastic counterpart, i.e. the sLWTA Transformer. The results are summarised in Table 4.14. The stochastic LWTA Transformer appears to be superior to the deterministic, achieving a BLEU-4 score of more than 1.5 units higher than the latter. The superiority holds for all ranks of BLEU scores in both the test and validation sets.

4.4.4.3 Ablation study

Model size

Contemporary NLP Transformer networks have a tendency towards large size (Lan et al., 2019; Brown et al., 2020; Devlin et al., 2018), reaching depths that can approach 100 layers. Conversely, SL Transformers are commonly much smaller, often no more than 2 to 3 layers deep. Our experiments, conducted on the Elementary23-SLT dataset, aimed to determine the optimal depth for the proposed stochastic Transformer. The results, depicted in Table 4.15, demonstrate that a depth of 2 proves to be the optimal choice, as indicated by the highest BLEU-4 scores on both the test and validation(dev) sets. A depth of 1 delivered results that were lower but still close, while increasing the depth beyond 2 resulted in a

decline in performance of around 1 to 1.5 units.

Table 4.15: BLEU-4 Scores per model depth

Depth	Dev	Test
1	6.35	5.57
2	6.67	5.69
3	5.09	4.63

The embedding size is another size-related hyperparameter crucial in Deep NLP models. Table 4.16 presents a study of the effect of different embedding sizes on the performance of the LWTA-Transformer on the Elementary23 dataset. The results indicate that an embedding size of 256 provides optimal performance. Smaller sizes appear insufficient to handle the complexity of the task, as they yielding the much lower scores of 4.39/4.07 for size=128. On the other hand, larger sizes, such as 512, do not improve the results.

The effect of Competing Units per Block

As previously discussed, the results of our experiments on the sLWTA Sign Language Transformer (Table 4.14) render it a superior solution for the Greek SL translation task compared to the deterministic model. A central aspect of this network is the use of the sophisticated LWTA layer, as opposed to a typical activation function such as Relu. The size of the competition blocks U is the main tunable hyperparameter of this technique. Through an examination of the commonly used sizes, presented in Table 4.17, we concluded that the most suitable choice for our case is $U = 2$, as suggested in (Panousis et al., 2019a). Larger sizes of $U = 4$ and $U = 8$ resulted in decreases of 0.22 and 0.76 BLEU-4 units, respectively; this is likely due to the high sparsity of the representations obtained from

Table 4.16: BLEU-4 Comparison between embedding sizes

Embedding Size	Dev	Test
128	4.39	4.07
256	6.67	5.69
512	5.32	5.27

Table 4.17: The effect of LWTA block size U

Competing Units (U)	Dev	Test
2	6.67	5.69
4	5.41	5.47
8	5.23	4.93

the LWTA blocks.

4.4.5 Discussion and Benchmarking

We now proceed with a direct evaluation of the translation accuracy attained on the Elementary23 dataset, and compare it to the results reported on other benchmark datasets. Table 4.18 summarises the achieved BLEU-4 scores for each case, covering both the main benchmarks and a curated selection of significant yet less directly comparable supplementary non-European datasets.

Table 4.18: Benchmarking BLEU-4 scores

Dataset	Dev	Test
Elementary23-SLT (Voskou et al., 2023)	6.67	5.69
Phoenix2014T (Voskou et al., 2021)	23.23	23.65
SWISSTXT-NEWS (Camgöz et al., 2021)	0.46	0.41
VRT-NEWS (Camgöz et al., 2021)	0.45	0.36
OpenASL (Shi et al., 2022)	6.57	6.72
CSL-Daily (Zhou et al., 2021)	20.80	21.34

As indicated in this table, using our method we have reported BLEU-4 scores reaching as high as 23.23/23.65 for the validation and test sets of Phoenix2014T, validating its standing as the highest-performing dataset. However, as previously noted, this dataset does come with limitations such as a restricted vocabulary, a narrowly focused topic, and a stringent structure. Analogous results emerge from applications on the CSL-Daily dataset. More specifically, researchers report impressive scores of 20.80/21.34 on this popular Chinese-SL dataset, which also bears similar constraints on vocabulary and content to Phoenix2014T. While these attributes enhance BLEU-4 performance, they diminish the

applicability of the developed SLT models for real-world users. In contrast, the objective of our work is to amplify the effectiveness of end-to-end SLT systems in genuine usage scenarios. These considerations make Phoenix2014T an imperfect comparison to the Elementary23 dataset, which boasts a more realistic design.

Conversely, models trained on SWISSTXT-NEWS and VRT-NEWS exhibit weak performance, with BLEU-4 scores < 1. The authors report scores of 0.46/0.41 and 0.45/0.36, respectively, which are considerably lower than the 6.67/5.69 achieved in our proposed subset. Unfortunately, with such poor scores, these datasets cannot provide any practical value, nor can they be regarded as a reliable benchmark for future SLT models; these facts are despite their extensive topic coverage and vast vocabulary size.

These contrasting outcomes highlight the significance of our SLT engine, the Stochastic LWTA-Transformer, trained using the Elementary23 dataset. Although the achieved BLEU-4 scores, approximately 6 units, are modest in comparison to state-of-the-art NLP models trained on large text corpora, they exhibit concrete translation capabilities. Thus, in contrast to previously mentioned SLT works and datasets, our model and dataset uniquely merge quantifiable outcomes with a wide and realistic thematic range. These attributes lend our novel dataset a distinct importance as a benchmark for future SLT research, and confer on our model the highly valuable capacity for real-world applications with socially impactful benefits. Lastly, we must note that recent efforts on the OpenASL dataset represent the sole instance that parallels our work, blending commendable results with quality content.

4.4.5.1 Qualitative Results

The quality of the automatic translations produced by our models varies; this is shown in Table 4.19, where three representative examples are presented. In some cases, such as the first example, the results are impressively accurate, with only minor numerical or grammatical errors. The next case belongs to a second category in which the context is partially captured, but the syntax deviates from the target. The final example represents the third group, where the model completely fails to detect any of the signed signals.

Table 4.19: Reference(R), Prediction(P), Translated Reference(Rt), Translated Prediction(Pt)

#	Translation Reference and Prediction
1	**R:** έχει 10 νομίσματα πόσα είναι τα χρήματά του συνολικά **P:** έχει 4 νομίσματα πόσα είναι τα χρήματά του συνολικά **Rt:** he has 10 coins how much is his money in total **Pt:** he has 4 coins how much is his money in total
2	**R:** συμπληρώνω τους αριθμούς που λείπουν στους πίνακες **P:** υπολογίζω και γράφω τους αριθμούς που λείπουν **Rt:** I fill in the missing numbers in the tables **Pt:** calculate and write the missing numbers
3	**R:** στο σχολείο μαθαίνουμε καινούρια πράγματα **P:** το περιβάλλον μου **Rt:** at school we learn new things **Pt:** my environment

4.4.6 Applications

As stated earlier, one of the primary goals of our work is to facilitate the use of automatic sign language translation engines in practical and real-world applications. Indeed, the proposed Sign Language Translation (SLT) method has already been implemented in a real-world setting, specifically within an educational platform. Targeting potential students who are interested in learning sign language, our model acts as an auxiliary tool, allowing for quick and straightforward verification through direct feedback at the sentence level for the signs performed.

Users are directed to a specialized interface where they can record themselves performing a sentence or phrase in the sign language being studied. This recording is then transmitted to the server to generate the corresponding text transcription. Both the initial recording and the transcription are made available to the user, serving as a tool for easy feedback on the learned material. Additionally, users can create multiple recordings of different signed sequences, with the ability to retain and review their previous recordings.

4.5 Conclusions

We proposed a novel Sign Language Translation (SLT) method with several notable advantages: (i) It does not require gloss sequences for training; (ii) It achieves a state-of-the-art BLEU-4 score on the PHOENIX 2014T dataset, rivaling methods that require gloss sequences and/or multiple data streams; and (iii) It has at least 70% lower memory requirements than previous state-of-the-art models. This was accomplished by designing a doubly stochastic Transformer network that: (i) replaces ReLU layers with stochastically competing linear units; and (ii) applies variational Bayesian inference to all connection weights across the entire network.

With an adapted version of the proposed stochastic Sign Language Transformer model, which uses a trajectory stream consisting of three channels: hand shapes, body poses, and facial expressions, we successfully developed an entirely gloss-free, multi-topic SLT method. This approach has clear real-world applicability and represents the first comprehensive SLT model for Greek Sign Language. This accomplishment was made possible through the introduction of a new Sign Language dataset, Elementary23. This is the largest and most complete collection of Greek Sign Language data to date. The dataset was collected with the help of external sign language linguist associates, ensuring excellent quality. Using a relatively simple yet particularly effective selection process, we created an SLT-compatible subset that meets established formatting standards and retains the desired features and quality standards. By comparing our results with those from other datasets and conducting related qualitative analyses, we demonstrated the robustness and value of our approach.

Chapter 5

Tabular Data Modelling

Despite the prevalence and significance of tabular data across numerous industries and fields, it has been relatively underexplored in the realm of deep learning. Even today, neural networks are often overshadowed by techniques such as gradient-boosted decision trees (GBDT). However, recent models are beginning to close this gap, outperforming GBDT in various setups and tasks and garnering increased attention in the field.

In this chapter drawing from this inspiration, we explore the application of stochastic deep nerual networks and we introduce a novel deep learning model specifically designed for tabular data. The foundation of this model is a Transformer-based architecture, carefully adapted to cater to the unique properties of tabular data through strategic architectural modifications such as the introduction of an attention bias and staticly connected unit augmentaly to the the core self-attention operations. Furhermore the predictive power of the model is based on the injection of two forms of stochasticity both in the concept of stochastic competition. First, we employ the "Local Winner Takes All" mechanism as the core building module. Second, we introduce a novel embedding layer that blends multiple linear embedding layers through a form of stochastic competition.

Model effectiveness is validated on a variety of widely-used, publicly available datasets. We show that, through incorporation of the said architectual modfification and stochastic elements, we yield state-of-the-art performance and mark a significant advancement in applying deep learning to tabular data.

5.1 Tabular Data

Tabular data is a fundamental and arguably one of the most commonly used formats in the fields of data science and machine learning. It is structured with rows and columns that represent individual observations and their corresponding features; this creates a simple two-dimensional, table-like body. Within it, various data types can be included. It frequently occurs and enjoys widespread popularity in sectors like healthcare, finance, and sciences because of its organizational clarity and its close ties with relational databases and spreadsheets. Yet, despite the prevalence and the seeming stractual simplicity of this layout, effectively modeling tabular data for common tasks like regression or classification continues to pose significant challenges.

In a tabular dataset, features can exhibit a wide range of characteristics and different formats, spanning from numerical values to more complex structures. This variation underscores the adaptability of tabular data to represent diverse information types within its structure. For the purposes of modeling, this diversity is streamlined, categorizing features into two primary types: continuous, represented by real numbers, and categorical, represented by discrete variables. Continuous features, which maintain uniformity within their respective columns, encapsulate quantitative measurements that vary across a spectrum, such as weights or heights. Categorical features, reflecting the dataset's capacity to encompass a broad array of data types, signify distinct categories or classifications, like product categories or nationalities, and are commonly encoded as integers to simplify computational processing.

Formally, a row in a tabular dataset of length s can be expressed as $\boldsymbol{x} \in \mathbb{R}^{s_r} \times \mathbb{N}^{s_n}$, where $s = s_r + s_n$. Here, s_r signifies the number of continuous features, illustrating the consistency within these columns by ensuring all values are of a real-number type. Conversely, s_n indicates the count of categorical features, showcasing the dataset's ability to integrate varied qualitative data types. This formal depiction not only simplifies the inherent complexity of tabular data but also illuminates the delicate balance between homogeneity within columns and heterogeneity across the dataset, allowing for a comprehensive

representation of data crucial for sophisticated analysis and modeling.

A central characteristic of tabular data is the assumption of no inherent relations between features. This stands in stark contrast to data types such as images or text, where geometrical and dynamical properties hold significance. The value or importance of a pixel in an image or a word in a text can be substantially influenced by its neighbors. This lack of spatial or sequential context in tabular data necessitates different analytical approaches and algorithms for effective interpretation and processing. Even though correlations may be present among tabular features, they are generally presumed to be unknown prior to modeling. Moreover, unlike in other data modalities, the positioning of features in a tabular row holds no intrinsic geometrical meaning.

5.2 Machnine Learning for Tabular Data

Numerous methodologies have been utilized historically to address relevelant tasks, ranging from basic statistical methods and elementary linear models to more advanced techniques such as support vector machines, among others. However, in recent years, preference has shifted towards tree-based models like Random Forests (Ho, 1995) and, notably, the much more advanced Gradient Boosted Decision Trees (GBDTs) (Friedman, 2002). selected for their superior performance. Deep Learning, a paradigm that has profoundly transformed learning across various data forms, has not yet been established as the primary approach for tabular data. Nevertheless, this trend is slowly changing. The recent emergence of novel deep learning models that outperform GBDTs on a spectrum of tabular datasets signifies a potential shift. Although the volume of related publications remains limited, the encouraging outcomes of these studies suggest a possible redefinition of the prevailing methodologies for tabular data analysis, potentially positioning deep learning at the forefront.

Gradient Boosted Decision Trees

As previously outlined, the most established methods in Tabular Data Modeling (TDM) currently belong to the family of tree-based algorithms, especially in the form of Gradient Boosted Decision Trees (GBDT). These algorithms rely on an ensemble of weak learners, sequentially generated as corrections to the existing ensembles in a gradient-driven fashion. Let $F_m(\boldsymbol{x})$ be the model at iteration m, where $\boldsymbol{x}$ is the input vector. The model is updated as

$$F_{m+1}(\boldsymbol{x}) = F_m(\boldsymbol{x}) + \eta \cdot h_m(\boldsymbol{x}) \tag{5.1}$$

where η is the learning rate, and $h_m(\boldsymbol{x})$ is the weak learner (decision tree) added at the m-th iteration. The weak learner is chosen to minimize a loss function $L(y, F(\boldsymbol{x}))$, given the true value y and the model prediction $F(\boldsymbol{x})$. The gradient boosting process aims to find $h_m(\boldsymbol{x})$ that best fits the negative gradient of the loss function with respect to the model predictions, effectively performing gradient descent in the function space.

These algorithms have gained prominence in various fields, from finance to healthcare, due to their ability to provide high-performance models that can be easily used through high-level APIs. This accessibility has made GBDT and its derivatives particularly popular among developers and data scientists, underscoring their adaptability and robust performance across diverse tabular data challenges.

The variants of GBDT, such as Catboost(Prokhorenkova et al., 2018), XGBoost(Chen et al., 2015), NGboost(Duan et al., 2020), and LightBoost(Ke et al., 2017), introduce specific improvements to this framework. XGBoost, for instance, is renowned for its scalability and efficiency, making it a preferred choice for handling complex and large datasets. Catboost efficiently manages categorical data, thus simplifying the preprocessing steps required for model training. NGboost advances the prediction accuracy by focusing on the probabilistic prediction of target variables, while LightBoost optimizes for speed and efficiency through a histogram-based approach for faster training times.

Deep Learning and Transformers

Until the conclusion of the previous decade, the deep learning methodologies that were applied to Tabular Data primarily focused exclusively on multi-layer perceptrons and other basic architectures. However, the past few years have seen a significant surge in the development of more sophisticated neural network designs, which have delivered remarkable results. These advanced designs have embraced a variety of strategies, including those that mimic decision trees or other forms of weak learners; frequently, these innovations draw inspiration from Gradient Boosted Decision Trees (GBDT). Two seminal architectures that exemplify this innovative approach are the Neural Oblivious Decision Ensembles (NODE)(Popov et al., 2019) and GrowNet(Badirli et al., 2020), which have both made substantial contributions to the field.

While these methodologies have achieved commendable outcomes, the direction of recent research has increasingly leaned towards the adoption of Transformer-based architectures. A notable example of this trend is the TabTransformer(Huang et al., 2020), which utilizes the transformer architecture to process categorical tabular features effectively. It then integrates these processed features with a fully connected layer to manage the numerical features. The amalgamation of satisfactory results and a relatively lightweight structure has garnered some popularity for this model amongst researchers and practitioners.

Conversely, architectures such as TabNet(Arik &Pfister, 2021) leverage the full computational capabilities of the Transformer and its attention mechanism to produce robust results through an encoder-decoder framework. In many instances, they are capable of matching or even surpassing the state-of-the-art performance of GBDT models. However, this pioneering work was characterized by a significant increase in complexity and prolonged training and inference durations, which rendered it somewhat impractical for certain applications. The FtTransformer(Gorishniy et al., 2021) addresses these issues by adopting an encoder-only design that analyzes all features collectively and then post-projects individual categorical features into separate vector representations through a simple yet efficient linear embedding layer. This method has successfully combined state-of-the-art results with a significantly reduced complexity, especially when compared to TabNet and similar models.

Moreover, the SAINT(Somepalli et al., 2021) model introduces an innovative inter-row attention layer, enhancing the model's ability to process data beyond the traditional row-by-row approach.

While the primary focus of our research centers on the exploration of core architectural innovations, it is essential not to overlook the significant contributions made by studies on auxiliary techniques or isolated sub-components. In addition to proposing advanced network architectures, a considerable number of investigations have explored the implications associated with various attributes and settings fundamental to deep learning methodologies. Noteworthy studies in this regard include those on transfer learning (Rubachev et al., 2022; Iida et al., 2021), as well as diverse embedding approaches that have demonstrated the potential to achieve strong results even with relatively simple architectures such as MLP-PLR (Gorishniy et al., 2022). Exceptional contributions, such as those by (Kotelnikov et al., 2023), which utilizes denoising diffusion probabilistic models, and (Gorishniy et al., 2023), which incorporates retrieval augmentation strategies, stand out. The majority of the techniques proposed in these studies can be effectively integrated with existing or future network architectures, potentially enhancing their performance even further.

5.3 The proposed Tabular Hybrid Transformer with Stochastic Competition

5.3.1 Overview

In Figure 5.1, we provide a comprehensive overview of the proposed model, which employs a hybrid architecture grounded on an encoder-only Transformer. This foundational architecture is augmented with stochastic elements and additional structural modifications, which we discuss in greater detail later in this Section.

Our proposed adaptations do not obliterate the necessity for a specific input structure compatible with the standard Transformer encoder. To achieve compatibility with this structure, our first step is to adapt the original data format, defined in $\mathbb{R}^{s_r} \times \mathbb{N}^{s_n}$, to one

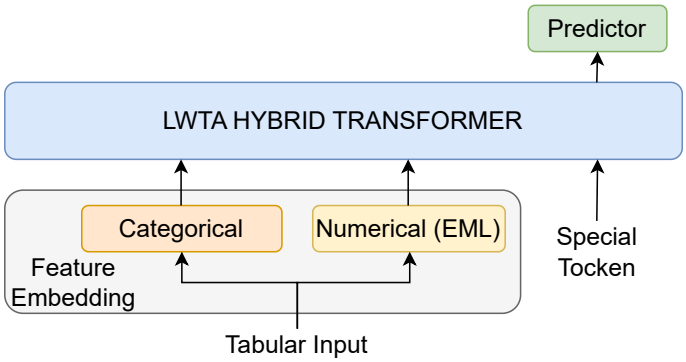

Figure 5.1: Overview of the proposed approach, exhibiting its core modules.

that fits the Transformer. Through embedding layers, each feature $x_i, i \in \{1, .., s\}$, be it numerical or categorical, is mapped onto a d-dimensional representation vector, given by $\boldsymbol{h}_i \in \mathbb{R}^d$. Eventually, a given input $\boldsymbol{x} = (\boldsymbol{x}_i)_{i=1}^s$ is mapped to a vector sequence $\boldsymbol{h} \in \mathbb{R}^{d \cdot s}$. Alongside this representation, we also add a vector, $\boldsymbol{h}_{special} \in \mathbb{R}^d$, that corresponds to an artificial "special token" with a static value. The terminal representation of this token is fed to a final regression or classification head, depending on the task at hand.

While our architectural design shares similarities with usual Transformers and preceding models on tabular data and other domains, it distinguishes itself through three key innovations that enhance its predictive capability: i) The adoption of the sophisticated stochastic LWTA layer (Panousis et al., 2019a). The latter while has been shown to yield improved results in a wide range of applications; yet, it has never been employed to networks designed for Tabular Data. ii) The introduction of a novel data-driven probabilistic selection among alternative (linear) feature embeddings. This enhancement adds an extra element of stochasticity and promotes richer feature representations. iii) The introduction of the Hybrid Transformer module, which is specifically designed for tabular data. This module merges the core Transformer encoder layer architecture with a parallel fully connected aggregation module. Tailored to capitalize on the static structure of tabular data, this aggregation module works by projecting the hidden representations back to scalar values and processing the aggregate result.

In the following subsections, we elaborate on each of the core novel elements that compose our Transformer-based approach.

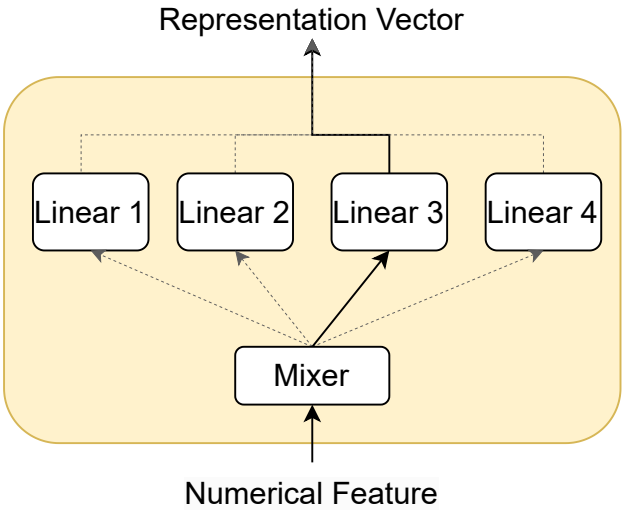

Figure 5.2: Illustration of the embedding mixture layer

5.3.2 Local Winner Takes All

The Stochastic "Local Winner Takes All" (LWTA) layers are adapted in similar way to our previous SL examples forming the main block of the model. The only difference is that in contrast to the Sign Langugage Translation models we now avoid the usage of variational weights. Although this does not affect the core competition engine of LWTA.

5.3.3 Feature Embedding - Embedding Mixture Layer

Feature embedding serves as a pivotal element in models like the one we propose, acting as the bedrock upon which later processing stages are built. In our approach, each categorical feature is separately processed via a standard linear embedding layer. This technique is stable and well-grounded in the literature, sharing conceptual similarities with word embedding commonly used in NLP.

Embedding of continuous values is much underexplored. Earlier work (Gorishniy et al., 2021) has mostly been limited to simple linear projections, computed independently for each feature. Recently, non-linear approaches have been explored and proved to be beneficial to the predictive accuracy (Gorishniy et al., 2022). In this work, we progress one step further, proposing a novel stochastic embedding layer that improves the expressive power of the vanilla approach. In our proposed method, instead of having a single pair

of weight and bias vectors, we use a set of J such pairs, defining J alternative (linear) embeddings, each indicated by an indicator j. To gain the representation vector of a continuous input, x_i, the model has now to select one of the so-defined alternative linear projections. It does so in a stochastic manner, where the probability of one alternative embedding being selected is driven by the value of x_i via (5.2); this selection rationale is illustrated in Figure 5.3 (left side).

$$f_{emb}(x_i) = x_i \boldsymbol{w}_j + \boldsymbol{b}_j, \qquad j \sim \boldsymbol{P}(\cdot|x_i, \boldsymbol{\theta}_w, \boldsymbol{\theta}_b) \tag{5.2}$$

where the posterior probability distribution over the linear mapping reads

$$\boldsymbol{P}(j|x, \boldsymbol{\theta}_w, \boldsymbol{\theta}_b) = \frac{e^{t_j}}{\sum_{j=1}^{J} e^{t_j}}, \qquad \boldsymbol{t} = x_i \cdot \boldsymbol{\theta}_w + \boldsymbol{\theta}_b \tag{5.3}$$

with $\boldsymbol{\theta}_w, \boldsymbol{\theta}_b \in \mathbb{R}^J$ denoting the trainable parameters directly involved in the selection process.

This embedding selection scheme can be described as a sort of competition among sub-parts at the embedding layer level; each competitor aims to dominate a broader range of input values. We posit that, in this way, the embedding engine can produce representations that are significantly richer than a single linear mapping. The eventually obtained embedding vector can vary considerably more than vanilla embeddings, based on the value regions of the input feature; this may allow for the identification of behavioral changes and shifts in statistical importance related to that feature. Additionally, the induced probabilistic transitions between different linear embedding enhance accuracy in uncertain areas of mapping, and also help reduce the risk of over-fitting.

While gating networks could be used to perform selection among alternative embeddings, our proposed method relies on competition, similar to stochastic LWTA layers althought in global layer to to layer competition rather than lwta's in local single neuron competition. This is an effective alternative that obviates the need to introduce more trainable parameters for the gating function, and the associated computational burden. Similarly, during training, $\boldsymbol{j}_i$ that corresponds to the embedding selection indicator vector of the numerical feature

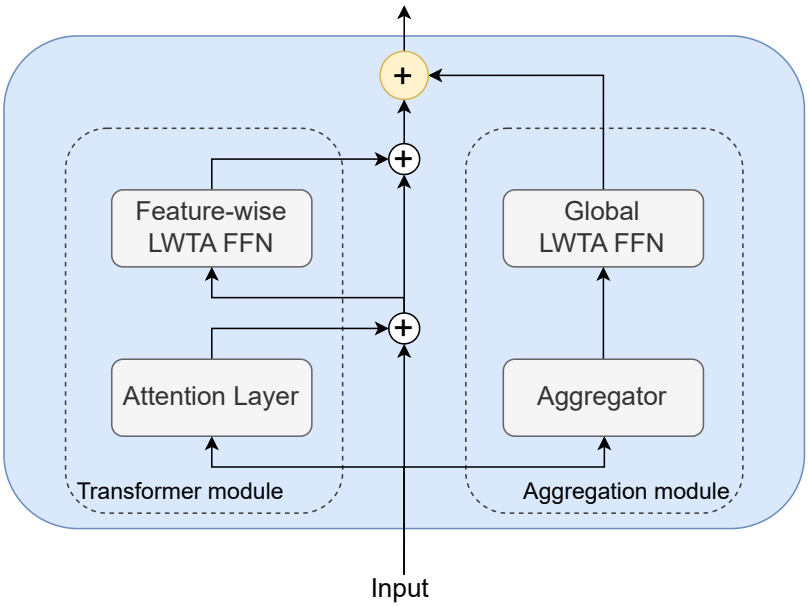

Figure 5.3: The hybrid Transformer module.

x_i is approximated via Gumbel-Softmax, to ensure effectiveness and stability:

$$\begin{aligned} \boldsymbol{j}_i &= \frac{\exp\left((\boldsymbol{\eta}_i + \boldsymbol{g}_i)/T\right)}{\sum_{j=1}^{U} exp((\eta_{i,j} + g_{i,j})/T)} \\ \boldsymbol{g} &= -\log(-\log \boldsymbol{z}),\ \boldsymbol{z} \sim \mathrm{U}(\mathbf{0}, \mathbf{1}) \end{aligned} \tag{5.4}$$

where $\boldsymbol{\eta}_i = \boldsymbol{w}_i x + \boldsymbol{b}_i$, and T the temperature hyperparameter.

5.3.4 Hybrid Transformer module

Typical Transformer input modalities, like text and videos, frequently display dimensionality that is subject to change, such as sentence lengths or video duration. Conversely, tabular datasets exhibit fixed, predefined dimensions. This distinct property offers an avenue for integrating static elements into the network, which would be unattainable in dynamically changing contexts.

Our so-obtained hybrid Transformer layer melds two essential subcomponents. Similar to a standard Transformer encoder layer, the first component is a feature-wise sequential arrangement of a Self-Attention layer and a Fully-Connected layer. In our work, we enhance the attention dot-product with a bias term, an adjustment we have empirically found

to be subtle yet effective. While the incorporation of various types of bias in attention has been previously explored, such as in (Dufter et al., 2022) where it was used to add relative positional information, our application is the first aimed at leveraging the structural properties of tabular data. The second component, a novel aspect of our design, is a *parallel module*. This module can technically be described as an LWTA-based global feedforward layer, as illustrated in Figure 5.3 (Right part). This innovation is inspired by previous research (Gorishniy et al., 2021; Kadra et al., 2021; Shwartz-Ziv &Armon, 2022) showing that despite the popularity of Transformer architectures, fully connected architectures can still yield remarkable results and should not be overlooked. Our hybrid approach facilitates an effective blend of static and dynamic feature interactions, contrasting with the purely dynamic nature of typical Transformers. Through this modification, we enhance the model's predictive capability with a small increase in computational cost.

The new module is presented with the d-dimensional embedding of each of the s input features, reprojects them onto scalar values and aggregates them into a single s-dimensional representation vector through the operation $\Phi : \mathbb{R}^{d \cdot s} \rightarrow \mathbb{R}^{s}$:

$$\Phi(\mathbf{h}) = (\boldsymbol{w}_i \cdot \boldsymbol{h}_i + b_i)_{i=1}^{s} \text{ where } \boldsymbol{w}_i, \boldsymbol{h}_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \tag{5.5}$$

The so obtained vector, $\Phi(\mathbf{h})$, is presented to a subsequent LWTA layer, followed by a Linear layer; this yields an output vector $\boldsymbol{z} \in \mathbb{R}^d$. The output from this module is incorporated into the representation of the special token, in an additive (residual) manner.

5.3.5 Training and Inference

The training objective of our proposed model is formulated as follows:

$$\mathcal{L}(\phi) = \mathbb{E}_{q(\cdot)}\big[\log p(\mathcal{D}|\{\phi\})\big] - \mathrm{KL}\big[Q(\{\boldsymbol{\xi}\})||P(\{\boldsymbol{\xi}\})\big] - \mathrm{KL}\big[Q(\{\boldsymbol{j}\})||P(\{\boldsymbol{j}\})\big] \tag{5.6}$$

where $\{\boldsymbol{\xi}\}$ the set of the LWTA winner indicators, $\{\boldsymbol{j}\}$ the embedding selection indicators and $\{\phi\}$ represents all the trainable parameters. It is captured by a composite functional consisting of three terms. The first term corresponds to the primary objective. It

incorporates the standard crossentropy loss for classification tasks and the mean squared loss for regression scenario. In both cases, the latent indicator vectors $\boldsymbol{\xi}$ and $\boldsymbol{j}$ are replaced by a differentiable (reparameterized) expression obtained through the Gumbel-Softmax trick. The second term encapsulates the Kullback-Leibler divergences between the posteriors and the priors of the winner indicators, using a uniform discrete prior distribution U:

$$\mathrm{KL}[Q(\boldsymbol{\xi})||P(\boldsymbol{\xi})] = \sum_{\forall \boldsymbol{\xi}} \sum_{i=1}^{U} Q(\xi_i) \log \left(Q(\xi_i)/U_i\right) \tag{5.7}$$

The third term is similar to the second, but quantifies the KL divergence between the posterior of embedding selection and a uniform discrete prior.

For model evaluation and inference, predictions are gained via Bayesian averaging. By executing the model multiple times, we average the resultant outputs from the employed classification or regression head.

5.4 Experimental Results

5.4.1 Benchmarking datasets

Table 5.1: Key statistics and properties of benchmarking datasets.

	HI	**AD**	**OT**	**HE**	**JA**	**YE**	**DI**	**HO**
Total Entries	98049	48842	61878	65196	83733	515345	53940	22784
Total Features	28	14	93	27	54	90	9	16
Catg Features	0	8	0	0	0	0	3	0
Task	C	C	C	C	C	R	R	R
Classes	2	2	9	100	4	–	–	–

In the experimental section, we employ 8 publicly available tabular datasets, in the same form as previously utilized in analogous research, such as (Gorishniy et al., 2021), and (Gorishniy et al., 2023). We use exactly the same train-validation-test split to facilitate fair comparison. Specifically, our analysis involves two datasets for binary classification, namely Higgs Small(HI) and Adult(AD); three datasets designed for multi-

class classification, namely Otto Group Products(OT) with nine classes, Helena(HE) with 100 classes, and Jannis(JA) with four classes; and three datasets tailored to regression tasks, namely Year Prediction(YE), Dimanond(DI), and House16H(HO). As reference metrics, we follow a common practice and use MSE for Regression and Accuracy for Classification Tasks. The bulk of the selected datasets are medium-sized, with row counts ranging from 20,000 to 100,000. However, to also examine how performance changes when using a significantly larger dataset, we also use Year Prediction, a particularly popular dataset encompassing around half a million features. In the context of feature types, the majority of datasets include numerical attributes, with feature dimensions ranging between 5 and 93. Exceptions to this pattern are the Adult and Diamond datasets, which additionally incorporate categorical features. Detailed analysis of data statistics is provided in Table 5.1.

5.4.2 Experimental setup

In all experiments, the AdamW (Loshchilov &Hutter, 2017) optimization algorithm was selected, with a small weight decay rate, $wd \leq 10^{-4}$. Training was divided into two sequential phases: a short warm-up featuring an ascending learning rate, and a subsequent main training part. In the latter phase, the learning rate commenced at $lr = 10^{-3}$ and was subject to a 50% reduction upon reaching a performance plateau. Additional hyper-parameters are an 8 head multi-head attention; an mc-dropout rate of $p = 0.1 - 0.25$; and a Gumbel Softmax temperature $T = 0.69$ for training and $T = 0.01$ for inference. As usual with Gumbel-Softmax reparameterization, it suffices that we consider sample size $N = 1$ for training; we draw $N = 64$ samples for inference. For input data prepossessing, appropriate normalization/scaling was employed, except for the OT dataset where original scaling was retained as suggested in (Gorishniy et al., 2021). Additionally, we re-scale the labels of HO and DI by a factor of 10^{-4} and 10^2, respectively for better illustration purposes.

The majority of the experiments were conducted on a single 16GB GPU (NVIDIA Quadro P5000). An exception to this setup was made for experiments on the Year

Table 5.2: Results comparison with related Deep Neural Networks.

Model	Classification(Acc ↑)					Regression(MSE ↓)			avg.
	HI	**AD**	**OT**	**HE**	**JA**	**YE**	**DI**	**HO**	rank
MLP	71.9%	85.3%	81.6%	38.3%	71.9%	78.37	1.960	9.6845	4.1
MLP-PLR	72.9%	**87.0%**	81.9%	–	–	–	**1.800**	**9.339**	1.7
Node	72.6%	85.8%	–	35.9%	72.7%	76.40	–	–	3.8
FtTransformer	73.0%	85.9%	81.7%	39.1%	73.2%	78.40	–	10.480	3.0
Saint	72.9%	86.0%	81.2%	–	–	–	1.877	10.510	3.9
STab	**73.2%**	86.1%	**82.5%**	**39.4%**	**73.6%**	**76.10**	1.825	9.650	1.4

Prediction dataset, which necessitated the use of a larger GPU (NVIDIA A100) due to their increased computational requirements.

For each dataset employed, we trained and validated our model four times with different random seeds each time to gain a more comprehensive understanding of its performance. The main results reported are the averages of these four trainings, and we additionally discuss the standard deviation among the four scores. Moreover, all the ensemble scores presented are combinations of these four runs.

5.4.3 Results Discussion

Table 5.2 presents a comparative evaluation of our proposed model against leading deep-learning benchmarks, specifically MLP-PLR, NODE, FtTransformer, and SAINT, in addition to the results of a basic MLP network used as a baseline. To maintain a focused examination of architectural differences, we intentionally exclude methods that rely on transfer learning or data augmentation.

For the proposed model (STab), we employ our recommended hyperparameters obtained through a brief tuning procedure and empirical consideration, to be detailed later in this section. For established benchmarks, we cite results from existing literature as provided in (Gorishniy et al., 2021), (Rubachev et al., 2022), (Gorishniy et al., 2023), or (Somepalli et al., 2021). It is important to mention that all the reported third-party results are the outcomes of well-conducted hyperparameter tuning, typically more extensive than ours,

Table 5.3: Results Analysis

Model	HI	AD	OT	HE	JA	YE	DI	HO
Mean	73.2%	86.1%	82.5%	39.4%	73.6%	76.10	1.825	9.650
STD	0.15%	0.09%	0.20%	0.17%	0.15%	0.21	0.027	0.59
Best	73.4%	86.2%	82.7%	39.5%	73.7%	75.88	1.789	9.1

and the reported numbers have been verified. This approach conserves computational resources and ensures impartiality by relying on multi-party verification of performance metrics.

As illustrated in Table 5.2, all high-performing models exhibit similar performance levels, with none showing significant superiority; each remains relatively close to the baseline. This observation suggests that the datasets may be approaching the optimal results achievable, rendering further enhancements a challenging endeavor; even a marginal advantage could prove crucial, especially in competitive environments. Our model demonstrates superior performance, outperforming existing neural network architectures in 5 out of 8 evaluated benchmarks. Moreover, we achieved an average ranking of 1.4 across all datasets and in the 3 instances where we did not secure the top result, we still managed to obtain the second position. Finally, STab outperforms FtTransformer, the architecturally closest approach to ours, across all setups. These outcomes provide a strong indication of the impact of the proposed innovations.

The exceptions to STab's superiority are observed in the HO, AD, and DI datasets, where our model trails behind MLP-PLR. Notably, these exceptions are the datasets with the fewest features from the benchmark collection, just 9 to 16 features per row. In addition, DI and AD are also the only two that include categorical features. This highlights a specific advantage of our approach in handling datasets that have a larger number of features and are exclusively composed of numerical features.

In Table 5.3, we present the performance of our model over 4 random seeds. In addition to the mean value, we also report the standard deviation and the best out of the 4 scores. We observe that the deviation in scores across different random seeds is relatively low,

Table 5.4: Ensemble models results comparison with Deep Networks and Gradient Boosted Decision Trees

Model	Classification(Acc ↑)					Regression(MSE ↓)		
	HI	**AD**	**OT**	**HE**	**JA**	**YE**	**DI**	**HO**
XGBoost	72.6%	**87.2%**	83.0%	37.5%	72.1%	79.98	1.877	10.09
XGBoost_{ens}	72.8%	**87.2%**	**83.2%**	38.8%	72.4%	78.49	1.850	10.00
CATBoost	72.6%	87.1%	82.5%	38.5%	72.3%	78.98	1.796	9.720
CATBoost_{ens}	72.9%	**87.2%**	82.7%	38.5%	72.7%	78.11	**1.769**	9.645
LGBoost	72.7%	87.1%	82.5%	38.0%	72.1%	79.50	1.85	10.15
LGBoost_{ens}	72.8%	**87.2%**	82.7%	38.1%	72.3%	78.82	1.80	10.02
MLP-PLR_{ens}	73.5%	**87.2%**	82.2%	–	–	–	1.769	**8.958**
Node_{ens}	72.7%	86.0%	–	36.1%	73.0%	76.02	–	–
$\text{FtTransformer}_{ens}$	73.3%	86.0%	82.4%	39.8%	73.9%	76.51	–	10.17
STab_{ens}	**73.6%**	86.2%	**83.2%**	**40.0%**	**74.0%**	**75.60**	1.781	9.300

especially given the stochastic nature of our approach, with $\sigma^2 \geq 0.2\%$ for all classification tasks, ≈ 0.2 for YE, and ≈ 0.025 for DI; the exception is the HO dataset, where $\sigma^2 > 0.5$, which is high relative to its mean value, likely reflecting its highly noisy nature. In the same table, we also include the best achieved scores.

Beyond the main results of Table 5.2 in Table 5.4, we extend the comparison to include ensemble models as well as three established GBDT paradigms in both single and ensemble configurations. While our model's superiority persists in ensemble settings, the margin of lead narrows slightly. Gradient Boosting models in their ensemble form closely align with our results on the OT task, and CATBoost' s marginally outperform us on DI. In addition, our model seems to benefit slightly less from ensembling compared to some older deterministic deep networks, possibly due to its inference mechanism via Bayesian averaging. Nonetheless, the ensemble version of our model remains the state-of-the-art solution for the majority of the evaluated tasks.

In Table 5.5, we list the main hyperparameters of the proposed model for each dataset, corresponding to the experimental results presented. These values might not showcase the absolute best performance, as we opted against exhaustive optimization and did not use black-box optimization techniques such as Optuna (Akiba et al., 2019) or Hyperopt

Table 5.5: Suggested main hyperparameters

Model	HI	AD	OT	HE	JA	YE	DI	HO
Dropout	0.25	0.1	0.25	0.25	0.25	0.25	0.1	0.125
Embedding Size	256	16	192	96	192	128	96	128
Depth	4	3	5	7	4	6	4	4

Table 5.6: Ablation study on different model variants.

Transformer Variant	Classification(Acc ↑)					Regression(MSE ↓)		
	HI	AD	OT	HE	JA	YE	DI	HO
Vanilla	73.0%	85.9%	81.7%	39.1%	73.2%	78.40	1.89	10.48
Stochastic	73.1%	86.0%	82.1%	39.3%	73.4%	77.05	1.84	10.02
Hybrid	**73.2%**	86.0%	81.9%	39.0%	73.2%	76.75	1.87	10.08
Full-model	**73.2%**	**86.1%**	**82.5%**	**39.4%**	**73.6%**	**76.10**	**1.83**	**9.65**

(Bergstra et al., 2013), which may require hundreds of iterations to provide optimal results. Additionally, in situations with marginally differing results, factors such as model size were also taken into consideration. Beyond these hyperparameters, a fixed LWTA block size $U = 2$ is employed, as suggested by the majority of related literature (Panousis et al., 2019a; Kalais &Chatzis, 2022), as well as a fixed $J = 16$ for the embedding mixture layer, which appears to be a well-performing value for most cases, both supported by preliminary analyses provided in the next section.

5.4.4 Ablation study

In this section we proceed into a deeper analysis of the impact of the novelties that are coverd by our model and the effect of the key controlling hyperparemeters we have introduced.

On the impact of the proposed modules.

In Table 5.6, we present a comparative analysis between variants of our approach, aiming to examine the impact of each proposed element. The vanilla variant is a regular transformer

encoder that incorporates neither the task-specific architectural modifications (the parallel module and the attention bias) nor the stochastic competition elements (embedding mixture and LWTA); this is equivalent to the FtTransformer model. The subsequent two cases correspond to the implementation of only the stochastic competition modules (Stochastic) and the use of the Hybrid Transformer Layer in a deterministic setup, similar to the vanilla version(Hybrid). Finally, we include the results of the full model to facilitate easier comparison.

Upon examination, it is evident that both the Stochastic and Hybrid variants exhibit performance enhancements over the Vanilla model; these enhancements have been obtained independently but the cumulative effect is more prominent when combined under the full model configuration. However, notable exceptions exist, such as in the case of the JA and HE datasets, where the Hybrid architecture, when applied in isolation, either fails to offer any benefit or reduces the performance. Similarly, for the HI dataset, the full model does not manifest any marked advantage over the deterministic hybrid framework.

Analysing the mixture embedding parameter J

Table 5.7: Targeted study on the the effect of mixture embedding parameter J

	HI(↑)	**HE(↑)**	**DI(↓)**	**HO(↓)**
J= 64	73.2%	39.4%	1.84	9.94
J= 16	73.2%	39.5%	1.83	9.55
J= 4	73.3%	39.5%	1.84	9.67
J= 1	73.2%	39.1%	1.87	9.88

To evaluate the influence of the Probabilistic Embedding Mixture and the relevant parameter J (mixture components) on our model's performance, we conducted a specific study. The results are displayed in Table 5.7, focusing on the significance of the parameter J. Notably, setting $J = 1$ is equivalent to employing a standard linear embedding. Data from Table 5.7 suggest that, in many cases, $J = 16$ is the optimal value or closely approximates it. Moreover, there is a noticeable improvement compared to the standard linear numerical feature embedding. However, minor adjustments to J, whether below

or above the optimal, typically do not lead to significant shifts in performance metrics. This finding supports our recommendation of a fixed $J = 16$ for the main analysis, thus diminishing the need for further time-consuming tuning.

On larger LWTA block sizes U

Table 5.8: The effect of LWTA block size U.

	HI(↑)	**HE(↑)**	**DI(↓)**	**HO(↓)**
U= 4	73.1%	39.2%	1.83	9.51
U= 2	73.2%	39.5%	1.83	9.55

Similarly, to further justify our decision to maintain a constant LWTA block size of $U = 2$, beyond previous literature, we conducted a brief analysis on the impact of a larger block size ($U = 4$) Table 5.8. The findings indicate that increasing the block size usually results in either suboptimal performance or only a minimal impact. These results, in line with prior studies, further validate our selection of $U = 2$ as an effective default setting, reducing the incentive for additional exploration.

The computational cost of the hybrid module

Table 5.9: Percentage increase in parameters and training/inference time due to the hybrid layer

	OT	**HI**	**AD**	**HE**	**JA**	**YE**	**DI**	**HO**
Parameters	35.2%	34.9%	10.4%	26.5%	33.6%	31.9%	23.4%	27.4%
Training Time	35.5%	46.2%	40.9%	43.7%	38.0%	37.6%	45.2%	46.5%
Inference Time	0.8%	3.0%	9.0%	2.1%	0.4%	0.9%	3.5%	3.4%

The introduction of the proposed hybrid transformer layer has led to significant improvements in accuracy, though it comes with an increase in the number of trainable parameters. Table 5.9 details the additional parameters incurred by employing the hybrid architecture over a standard transformer encoder, as well as the corresponding rise in

training and inference times for a single batch. Although the increase in parameters is measurable, it is not excessive, with an average rise of 28% and a maximum of just over 35%. Furthermore, this increase moderately affects training time, with increments ranging from 35% to 47%. As expected, the augmentation does not significantly impact inference time, given that the additional modules operate in parallel. Finally, it is essential to note that, despite the additional parameters, our encoder-only architectures remain significantly more resource-efficient than some previous models, such as TabNet.

On the potential utilization of Variational weights

In contrast to the Sign Language Transformer method discussed in the preceding chapter, which utilized a Gaussian variational posterior for the trainable parameters, this Tabular model employs conventional point estimations for the weights. To justify this decision, we present a concise study in Table 5.10. The results suggest that this approach could potentially degrade the model's performance or leave it unchanged. This is likely due to the increased complexity and stochasticity introduced by this methodology. These observations, coupled with the doubling of parameter count, which in turn escalates training time and necessitates more resources—thereby requiring more expensive equipment for training—provide a solid justification against the adoption of such an approach.

Table 5.10: Scores of Gaussian vs deterministic weights

	HI(↑)	**HE(↑)**	**DI(↓)**	**HO(↓)**
Gaussian	73.1%	38.9%	1.84	9.80
Point Estimation	73.2%	39.4%	1.84	9.65

Bayesian Averaging and Sample Size

Given the stochastic characteristics of our model, we derive its final prediction via Bayesian averaging. Figure 5.4 illustrates the correlation between sample size and prediction quality. Consistent with expectations, an increase in sample size tends to enhance and stabilize

the predictions. A sample size of $N > 20$ seems to be a robust selection, as the prediction quality begins to plateau around this value.

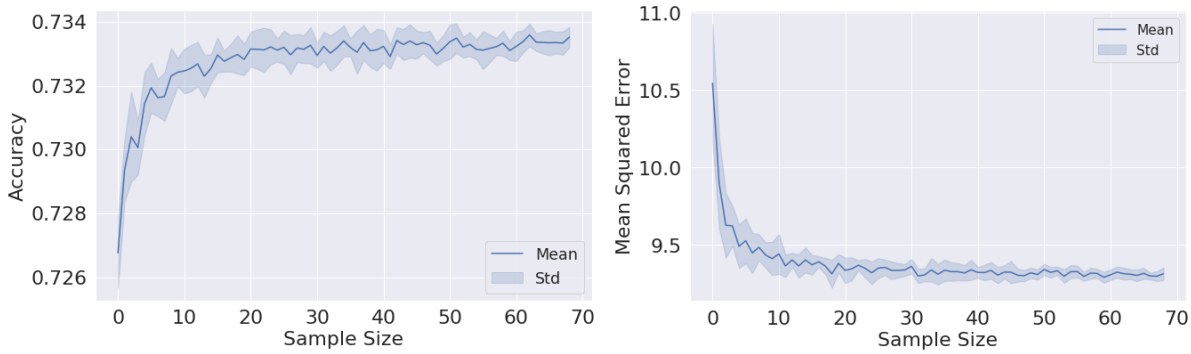

Figure 5.4: The effect of Sample size N on model's performance : (Left) Accuracy Higgs boson detection, (Rigth) Mean Squared Error on House16H

While our averaging approach may look similar to model ensembling, it's crucial to point out that they differ in key aspects. Unlike model ensembling, which requires training multiple (N) distinct models, our method needs just a single model to be trained. This means no need for extended training processes neither additional memory and storage space. Additionally, while it is true that inference time increases linearly with N in either case, this does not hold for single-row inference or small batches. In these cases, even for very large N, drawing N samples can be performed in parallel on a single GPU without additional delays. This is particularly advantageous for real-time applications requiring low latency and rapid response times.

5.5 Limitations

While the proposed model attained state-of-the-art results across the majority of tasks, it falls short of achieving universal superiority. However, it is important to acknowledge that no existing model has reached this level of performance to date. Furthermore, similarly to many older relevant deep networks, this model exhibits a complex architecture and necessitates significantly more resources for both training and inference compared to simpler models such as Gradient Boosting Decision Trees.

5.6 Conclusion

In this study, we introduce a novel approach to tabular data modelling by harnessing contemporary deep learning, with a particular emphasis on stochastic competition techniques. We employ a stochastic Transformer-based model with a modified task-adapted architecture. The model's computational prowess is further augmented by the integration of the stochastic LWTA layer. Additionally, we unveil a distinctive embedding mixture layer for numerical features, seamlessly fusing multiple linear mappings through a stochastic competition mechanism. As a testament to our approach's efficacy, we secured state-of-the-art results on a majority of eight popular benchmarks and achieved second place among recent deep learning methodologies in the remaining instances. Notably, these advantages persist even in ensemble model configurations.

In upcoming research endeavors, we recommend a thorough exploration of stochastic competition methods, with the goal of enhancing model performance for tabular data and setting the stage for a deep learning framework in this GBDT-dominated area. Another avenue of interest is understanding how these stochastic techniques can leverage sample outcomes to estimate metrics beyond just expected values; this includes assessing uncertainties and probing into the distributional aspects of predictions. Also, incorporating advanced strategies, such as smart data augmentation, transfer learning, and meta-learning, offers a promising perspective for future studies. Historically, these methodologies have demonstrated their effectiveness by markedly improving model outcomes, suggesting their potential to elevate the efficacy of our proposed architecture.

Chapter 6

Conclusions and Future Work

In this thesis, we proposed and developed a stochastic Transformer paradigm, primarily based on stochastic competition. Specifically, we implemented a stochastic Local Winner Takes All approach as a central element in the Transformer's core. We effectively integrated this component with additional stochastic modules. For the sign language transformer, we employed a Bayesian variational treatment over the weights, utilizing Gaussian posteriors and applying a weight compression technique that leverages the uncertainty estimations. For the tabular transformer, we introduced a stochastic Embedding Mixture layer based on global rather than local feature competition. Additionally, we enhanced the standard transformer encoder architecture by adding a novel hybrid layer that takes advantage of the unique properties of this data format.

Using the proposed method, we successfully tackled the challenging task of sign language translation. We achieved state-of-the-art results on the most well-studied dataset in the field, surpassing previous works in terms of translation quality, reducing the dependency on gloss annotation, and simultaneously decreasing the memory needs of the network. Following this achievement, we made significant progress in automatic Greek Sign Language processing, especially in the crucial task of full end-to-end Greek sign language translation. This was made possible through the introduction of a high-quality, wide-domain dataset, the largest ever on Greek Sign Language, named Elementary23. By altering the feature extraction engine of our model, we managed to train a Greek SLT model that balances real-world applicability with respectable translation accuracy. The outcomes demonstrated that our model design is particularly effective for the particularly complex and data-scarce task of sign language translation.

The suggested paradigm was further validated in the context of tabular data modeling.

Specifically, we trained and evaluated our model on eight different tabular datasets, each with varied properties and commonly studied in relevant literature. We achieved state-of-the-art results on most of these datasets while maintaining competitive performance on the others. In addition to the transformer with local stochastic competition, we also proposed (i) a novel competition-based feature mixture embedding module and (ii) a hybrid transformer architecture. We demonstrated that each proposed component, when tested in isolation, enhances baseline results significantly, and their combined application synergistically maximizes their potential impact. These advances not only improved performance standards across several datasets; they also pushed the field of deep learning on tabular data forward, equipping researchers with new innovative tools while placing stochastic competition at the forefront of future research efforts.

The success of the proposed approach in two different and completely dissimilar tasks confirmed the effectiveness and potential of Deep Transformer Neural Networks with stochastic competition, supporting an indication for further exploration in new areas such as modern LLMs or in fundamental vision and time series applications. Additionally, future work may also extend to the examination of further SLT and tabular-related tasks.

In the context of sign language, as delineated in the manuscript, the model currently serves as an automated tool to assist learners of sign language by verifying their signing attempts. While this functionality is beneficial, a more focused approach could involve the development of sentence-level sign language verification models specifically designed to assess learning attempts, where the targeted sentences are predefined. Such models would directly evaluate the accuracy of the learners' signs, potentially simplifying the verification process and enhancing overall accuracy compared to indirect methods.

Concerning the pathway of tabular data, a logical progression involves utilizing stochasticity and the resulting distribution of predictions to facilitate precise uncertainty estimation. Although numerous techniques for this purpose have been previously suggested, the prevalence of modern high-performing models that incorporate this feature remains limited.

Bibliography

Agris, U. v. and Kraiss, K.-F. SIGNUM database: Video corpus for signer-independent continuous sign language recognition. In Dreuw, P., Efthimiou, E., Hanke, T., Johnston, T., Martínez Ruiz, G., and Schembri, A. (eds.), *Proceedings of the LREC2010 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies*, pp. 243–246, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL https://www.sign-lang.uni-hamburg.de/lrec/pub/10006.pdf.

Ahmed, W., Chanda, K., and Mitra, S. Vision based hand gesture recognition using dynamic time warping for indian sign language. In *2016 International Conference on Information Science (ICIS)*, pp. 120–125. IEEE, 2016.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Al-Rousan, M., Assaleh, K., and Tala'a, A. Video-based signer-independent arabic sign language recognition using hidden markov models. *Applied Soft Computing*, 9(3):990–999, 2009.

Albanie, S., Varol, G., Momeni, L., Bull, H., Afouras, T., Chowdhury, H., Fox, N., Woll, B., Cooper, R., McParland, A., et al. Bbc-oxford british sign language dataset. *arXiv preprint arXiv:2111.03635*, 2021.

Ameen, S. and Vadera, S. A convolutional neural network to classify american sign language fingerspelling from depth and colour images. *Expert Systems*, 34(3):e12197, 2017.

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer, 2021.

Athitsos, V. and Sclaroff, S. 3d hand pose estimation by finding appearance-based matches in a large database of training views. Technical report, Boston University Computer Science Department, 2001.

Athitsos, V. and Sclaroff, S. Estimating 3d hand pose from a cluttered image. Technical report, Boston University Computer Science Department, 2003.

Athitsos, V., Neidle, C., Sclaroff, S., Nash, J., Stefan, A., Yuan, Q., and Thangali, A. The american sign language lexicon video dataset. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2008. doi: 10.1109/CVPRW.2008.4563181.

Back, A. and Keith, W. Bayesian neural networks for financial asset forecasting, 2019.

Badirli, S., Liu, X., Xing, Z., Bhowmik, A., Doan, K., and Keerthi, S. S. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

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

Bellugi, U. and Fischer, S. A comparison of sign language and spoken language. *Cognition*, 1(2-3):173–200, 1972.

Bendarkar, D., Somase, P., Rebari, P., Paturkar, R., and Khan, A. Web based recognition and translation of american sign language with cnn and rnn. 2021.

Bergstra, J., Yamins, D., and Cox, D. D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123, 2013.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Bilal, S., Akmeliawati, R., Shafie, A. A., and El Salami, M. J. Modeling of human upper body for sign language recognition. In *The 5th International Conference on Automation, Robotics and Applications*, pp. 104–108. IEEE, 2011.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html.

Bragg, D., Koller, O., Bellard, M., Berke, L., Boudrealt, P., Braffort, A., Caselli, N., Huenerfauth, M., Kacorri, H., Verhoef, T., et al. Sign language recognition, generation, and translation: An interdisciplinary perspective. *arXiv preprint arXiv:1908.08597*, 2019.

Brand, M., Oliver, N., and Pentland, A. Coupled hidden markov models for complex action recognition. In *cvpr*, volume 97, pp. 994, 1997.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Camgöz, N. C., Kındıroğlu, A. A., Karabüklü, S., Kelepir, M., Özsoy, A. S., and Akarun, L. BosphorusSign: a Turkish sign language recognition corpus in health and finance domains. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1383–1388, 2016.

Camgoz, N. C., Hadfield, S., Koller, O., Ney, H., and Bowden, R. Neural sign language translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7784–7793, 2018. doi: 10.1109/CVPR.2018.00812.

Camgöz, N. C., Koller, O., Hadfield, S., and Bowden, R. Sign language transformers: Joint end-to-end sign language recognition and translation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10020–10030. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01004. URL https://doi.org/10.1109/CVPR42600.2020.01004.

Camgöz, N. C., Saunders, B., Rochette, G., Giovanelli, M., Inches, G., Nachtrab-Ribback, R., and Bowden, R. Content4all open research sign language translation datasets. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pp. 1–5. IEEE, 2021.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

Chai, X., Wanga, H., Zhoub, M., Wub, G., Lic, H., and Chena, X. Devisign: dataset and evaluation for 3d sign language recognition. *Technical report, Beijing, Tech. Rep*, 2015.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Cheok, M. J., Omar, Z., and Jaward, M. H. A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics*, 10 (1):131–153, 2019.

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

Cihan Camgoz, N., Hadfield, S., Koller, O., Ney, H., and Bowden, R. Neural sign language translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7784–7793, 2018.

Cooper, H., Holt, B., and Bowden, R. Sign language recognition. In *Visual Analysis of Humans*, pp. 539–562. Springer, 2011.

Cui, R., Liu, H., and Zhang, C. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7361–7369, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2018.

Dilsizian, M., Yanovich, P., Wang, S., Neidle, C., and Metaxas, D. N. A new framework for sign language recognition based on 3d handshape identification and linguistic modeling. In *LREC*, pp. 1924–1929, 2014.

Dilsizian, M., Tang, Z., Metaxas, D., Huenerfauth, M., and Neidle, C. The importance of 3d motion trajectories for computer-based sign recognition. In *Proceedings of the 7th Workshop on the Representation and Processing of Sign Languages: Corpus Mining, Language Resources and Evaluation Conference 2016*. European Language Resources Association (ELRA), 2016.

Ding, L. and Martinez, A. M. Recovering the linguistic components of the manual signs in american sign language. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 447–452. IEEE, 2007.

Ding, L. and Martinez, A. M. Modelling and recognition of the linguistic components in american sign language. *Image and vision computing*, 27(12):1826–1844, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dreuw, P., Ney, H., Martinez, G., Crasborn, O. A., Piater, J., Miguel Moya, J., and Wheatley, M. The signspeak project-bridging the gap between signers and speakers. 2010.

Duan, T., Anand, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A., and Schuler, A. Ngboost: Natural gradient boosting for probabilistic prediction. In *International conference on machine learning*, pp. 2690–2700. PMLR, 2020.

Duarte, A., Palaskar, S., Ventura, L., Ghadiyaram, D., DeHaan, K., Metze, F., Torres, J., and Giro-i Nieto, X. How2sign: a large-scale multimodal dataset for continuous american sign language. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2735–2744, 2021.

Dufter, P., Schmitt, M., and Schütze, H. Position Information in Transformers: An Overview. *Computational Linguistics*, 48(3):733–763, 2022.

Elliott, E. A. and Jacobs, A. M. Facial expressions, emotions, and sign languages. *Frontiers in psychology*, 4:115, 2013.

Elliott, R., Cooper, H., Glauert, J., Bowden, R., and Lefebvre-Albaret, F. Search-by-example in multilingual sign language databases. In *Proceedings of the Second International Workshop on Sign Language Translation and Avatar Technology (SLTAT)*, Dundee, Scotland, October 23 2011. URL http://personal.ee.surrey.ac.uk/Personal/H.Cooper/research/papers/SBE_SLTAT.pdf.

Elmezain, M., Al-Hamadi, A., and Michaelis, B. Real-time capable system for hand gesture recognition using hidden markov models in stereo color image sequences. 2008.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021.

Freitas, F. A., Peres, S. M., Lima, C. A. M., and Barbosa, F. V. Grammatical facial expression recognition in sign language discourse: a study at the syntax level. *Information Systems Frontiers*, 19(6):1243–1259, Dec 2017. ISSN 1572-9419. doi: 10.1007/s10796-017-9765-z. URL https://doi.org/10.1007/s10796-017-9765-z.

Friedman, J. H. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

Ghahramani, Z. and Jordan, M. I. Factorial hidden markov models. In *Advances in Neural Information Processing Systems*, pp. 472–478, 1996.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goldstein, N. E., Sexton, J., and Feldman, R. S. Encoding of facial expressions of emotion and knowledge of american sign language. *Journal of Applied Social Psychology*, 30(1): 67–76, 2000.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.

Gorishniy, Y., Rubachev, I., and Babenko, A. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35: 24991–25004, 2022.

Gorishniy, Y., Rubachev, I., Kartashev, N., Shlenskii, D., Kotelnikov, A., and Babenko, A. Tabr: Unlocking the power of retrieval-augmented tabular deep learning. *arXiv preprint arXiv:2307.14338*, 2023.

Guerra, R. R., Rezende, T. M., Guimarães, F. G., and Almeida, S. G. M. Facial expression analysis in brazilian sign language for sign recognition. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, pp. 216–227. SBC, 2018.

Guo, J., Lei, Z., Wan, J., Avots, E., Hajarolasvadi, N., Knyazev, B., Kuharenko, A., Junior, J. C. S. J., Baró, X., Demirel, H., et al. Dominant and complementary emotion recognition from still images of faces. *IEEE Access*, 6:26391–26403, 2018.

Hammer, J. H. and Beyerer, J. Robust hand tracking in realtime using a single head-mounted rgb camera. In Kurosu, M. (ed.), *Human-Computer Interaction. Interaction Modalities and Techniques*, pp. 252–261, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39330-3.

Hammer, J. H., Voit, M., and Beyerer, J. Motion segmentation and appearance change detection based 2d hand tracking. In *2016 19th International Conference on Information Fusion (FUSION)*, pp. 1743–1750. IEEE, 2016.

Han, J., Awad, G., and Sutherland, A. Modelling and segmenting subunits for sign language recognition based on hand motion analysis. *Pattern Recognition Letters*, 30(6): 623–633, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Heap, T. and Hogg, D. Towards 3d hand tracking using a deformable model. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 140–145. Ieee, 1996.

Heckerman, D. A tutorial on learning with bayesian networks. *Innovations in Bayesian networks*, pp. 33–82, 2008.

Ho, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pp. 278–282. IEEE, 1995.

Huang, J., Zhou, W., Li, H., and Li, W. Sign language recognition using 3d convolutional neural networks. In *2015 IEEE international conference on multimedia and expo (ICME)*, pp. 1–6. IEEE, 2015.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Hussain, S., Saxena, R., Han, X., Khan, J. A., and Shin, H. Hand gesture recognition using deep learning. In *2017 International SoC Design Conference (ISOCC)*, pp. 48–49. IEEE, 2017.

Iida, H., Thai, D., Manjunatha, V., and Iyyer, M. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*, 2021.

Isaacs, J. and Foo, S. Hand pose estimation for american sign language recognition. In *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*, pp. 132–136. IEEE, 2004.

Jain, A. K., Mao, J., and Mohiuddin, K. M. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax, 2017.

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. Hands-on bayesian neural networks – a tutorial for deep learning users, 2021.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. Well-tuned simple nets excel on tabular datasets. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 23928–23941. Curran Associates, Inc., 2021.

Kalais, K. and Chatzis, S. Stochastic deep networks with linear competing units for model-agnostic meta-learning. In *International Conference on Machine Learning*, pp. 10586–10597. PMLR, 2022.

Karami, A., Zanj, B., and Sarkaleh, A. K. Persian sign language (psl) recognition using wavelet transform and neural networks. *Expert Systems with Applications*, 38(3):2661–2667, 2011.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014a. URL http://arxiv.org/abs/1412.6980.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014b.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., and Jatakia, J. Human skin detection using rgb, HSV and ycbcr color models. *CoRR*, abs/1708.02694, 2017. URL http://arxiv.org/abs/1708.02694.

Koller, O., Ney, H., and Bowden, R. Deep learning of mouth shapes for sign language. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

Koller, O., Ney, H., and Bowden, R. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3793–3802, 2016a.

Koller, O., Zargaran, O., Ney, H., and Bowden, R. Deep sign: Hybrid cnn-hmm for continuous sign language recognition. In *Proceedings of the British Machine Vision Conference 2016*, 2016b.

Koller, O., Camgoz, N. C., Ney, H., and Bowden, R. Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2306–2320, 2019.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pp. 17564–17579. PMLR, 2023.

Kreutzer, J., Bastings, J., and Riezler, S. Joey NMT: A minimalist NMT toolkit for novices. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 109–114, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-3019. URL https://www.aclweb.org/anthology/D19-3019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kurakin, A., Zhang, Z., and Liu, Z. A real time system for dynamic hand gesture recognition with a depth sensor. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 1975–1979, 2012.

Kurdyumov, R., Ho, P., and Ng, J. Sign language classification using webcam images, 2011.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lee, M. W. and Cohen, I. Human upper body pose estimation in static images. In *European Conference on Computer Vision*, pp. 126–138. Springer, 2004.

Li, D., Rodriguez, C., Yu, X., and Li, H. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*, pp. 1459–1469, 2020.

Li, G., Tang, H., Sun, Y., Kong, J., Jiang, G., Jiang, D., Tao, B., Xu, S., and Liu, H. Hand gesture recognition based on convolution neural network. *Cluster Computing*, 22 (2):2719–2729, 2019.

Liang, Z.-j., Liao, S.-b., and Hu, B.-z. 3d convolutional neural networks for dynamic sign language recognition. *The Computer Journal*, 61(11):1724–1736, 2018.

Liao, Y., Xiong, P., Min, W., Min, W., and Lu, J. Dynamic sign language recognition based on video sequence with blstm-3d residual networks. *IEEE Access*, 7:38044–38054, 2019.

Liu, N. and Lovell, B. C. Gesture classification using hidden markov models and viterbi path counting. *Proceeding: VIIth Digital Image Computing Techniques and Applications, Sydney*, 2003.

Liu, T., Zhou, W., and Li, H. Sign language recognition with long short-term memory. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2871–2875. IEEE, 2016.

Liu, Z., Lin, J., Ning, Y., Cao, Y., Zhang, H., Dong, Z., Wei, F., and Guo, B. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning, 2017.

Lu, P. and Huenerfauth, M. Collecting and evaluating the cuny asl corpus for research on american sign language animation. *Computer Speech & Language*, 28(3):812–831, 2014. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2013.10.004. URL https://www.sciencedirect.com/science/article/pii/S0885230813000879.

Lu, S., Metaxas, D., Samaras, D., and Oliensis, J. Using multiple cues for hand tracking and model refinement. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pp. II–443. IEEE, 2003.

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

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. ICLR*, 2017.

Martínez, A. M., Wilbur, R. B., Shay, R., and Kak, A. C. Purdue rvl-slll asl database for automatic recognition of american sign language. In *ICMI*, pp. 167–172. IEEE Computer Society, 2002. ISBN 0-7695-1834-6. URL http://dblp.uni-trier.de/db/conf/icmi/icmi2002.html#MartinezWSK02.

Masood, S., Srivastava, A., Thuwal, H. C., and Ahmad, M. Real-time sign language gesture (word) recognition from video sequences using cnn and rnn. In *Intelligent Engineering Informatics*, pp. 623–632. Springer, 2018.

Mayberry, R. I. and Kluender, R. Rethinking the critical period for language: New insights into an old question from american sign language. *Bilingualism: Language and Cognition*, 21(5):886–905, 2018.

Mcculloch, W. and Pitts, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

McCullough, S., Emmorey, K., and Sereno, M. Neural organization for recognition of grammatical and emotional facial expressions in deaf asl signers and hearing nonsigners. *Cognitive Brain Research*, 22(2):193–203, 2005.

Medsker, L. R. and Jain, L. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Molchanov, P., Gupta, S., Kim, K., and Kautz, J. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–7, 2015.

Monir, S., Rubya, S., and Ferdous, H. S. Rotation and scale invariant posture recognition using microsoft kinect skeletal tracking feature. In *2012 12th international conference on intelligent systems design and applications (ISDA)*, pp. 404–409. IEEE, 2012.

Mukkamala, M. C. and Hein, M. Variants of rmsprop and adagrad with logarithmic regret bounds. *CoRR*, abs/1706.05507, 2017. URL http://arxiv.org/abs/1706.05507.

Murray, K. Lack of british sign language interpreters putting deaf people at risk. *The Guardian*, 8, 2013.

Nandy, A., Prasad, J. S., Mondal, S., Chakraborty, P., and Nandi, G. C. Recognition of isolated indian sign language gesture in real time. In Das, V. V., Vijayakumar, R., Debnath, N. C., Stephen, J., Meghanathan, N., Sankaranarayanan, S., Thankachan, P. M., Gaol, F. L., and Thankachan, N. (eds.), *Information Processing and Management*, pp. 102–107, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12214-9.

Nikam, A. S. and Ambekar, A. G. Sign language recognition using image based hand gesture recognition techniques. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pp. 1–5. IEEE, 2016.

Ong, E.-J., Cooper, H., Pugeault, N., and Bowden, R. Sign language recognition using sequential pattern trees. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Providence, Rhode Island, USA, June 16 – 21 2012. URL http://personal.ee.surrey.ac.uk/Personal/H.Cooper/research/papers/Ong_Sign_2012.pdf.

Oszust, M. and Wysocki, M. Modelling and recognition of signed expressions using subunits obtained by data–driven approach. In Ramsay, A. and Agre, G. (eds.), *Artificial Intelligence: Methodology, Systems, and Applications*, pp. 315–324, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33185-5.

Oszust, M. and Wysocki, M. Polish sign language words recognition with kinect. In *2013 6th International Conference on Human System Interactions (HSI)*, pp. 219–226, 2013. doi: 10.1109/HSI.2013.6577826.

Oyedotun, O. K. and Khashman, A. Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, 28(12):3941–3951, 2017.

Özdemir, O., Kındıroğlu, A. A., Cihan Camgoz, N., and Akarun, L. BosphorusSign22k Sign Language Recognition Dataset. In *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, 2020.

Panousis, K., Chatzis, S., and Theodoridis, S. Nonparametric bayesian deep networks with local competition. In *International Conference on Machine Learning*, pp. 4980–4988. PMLR, 2019a.

Panousis, K., Chatzis, S., and Theodoridis, S. Nonparametric Bayesian deep networks with local competition. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4980–4988. PMLR, 09–15 Jun 2019b. URL https://proceedings.mlr.press/v97/panousis19a.html.

Panousis, K., Chatzis, S., and Theodoridis, S. Stochastic local winner-takes-all networks enable profound adversarial robustness. In *Bayesian Deep Learning NeurIPS workshop*, 2021a. URL https://openreview.net/forum?id=CcSPRnm1M5s.

Panousis, K. P., Chatzis, S., Alexos, A., and Theodoridis, S. Local competition and stochasticity for adversarial robustness in deep learning, 2021b.

Papamakarios, G. Comparison of modern stochastic optimization algorithms. 2014.

Partaourides, H., Voskou, A., Kosmopoulos, D., Chatzis, S., and Metaxas, D. N. Variational bayesian sequence-to-sequence networks for memory-efficient sign language translation. In *International Symposium on Visual Computing*, pp. 251–262. Springer, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Popov, S., Morozov, S., and Babenko, A. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Pu, J., Zhou, W., and Li, H. Sign language recognition with multi-modal features. In *Pacific Rim Conference on Multimedia*, pp. 252–261. Springer, 2016a.

Pu, J., Zhou, W., Zhang, J., and Li, H. Sign language recognition based on trajectory modeling with hmms. In *International Conference on Multimedia Modeling*, pp. 686–697. Springer, 2016b.

Quiroga, F., Antonio, R., Ronchetti, F., Lanzarini, L. C., and Rosete, A. A study of convolutional architectures for handshape recognition applied to sign language. In *XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017).*, 2017.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Ravikiran, J., Mahesh, K., Mahishi, S., Dheeraj, R., Sudheender, S., and Pujari, N. V. Finger detection for sign language recognition. In *Proceedings of the international MultiConference of Engineers and Computer Scientists*, volume 1, pp. 18–20, 2009.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Ricco, S. and Tomasi, C. Fingerspelling recognition through classification of letter-to-letter transitions. In *Asian Conference on Computer Vision*, pp. 214–225. Springer, 2009.

Ronchetti, F., Quiroga, F., Estrebou, C., Lanzarini, L., and Rosete, A. Lsa64: A dataset of argentinian sign language. *XX II Congreso Argentino de Ciencias de la Computación (CACIC)*, 2016.

Rosales, R. and Sclaroff, S. Learning body pose via specialized maps. In *Advances in neural information processing systems*, pp. 1263–1270, 2002.

Rubachev, I., Alekberov, A., Gorishniy, Y., and Babenko, A. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

Ruder, S. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

Sandler, W. and Lillo-Martin, D. *Sign language and linguistic universals*. Cambridge University Press, 2006.

Senghas, A. and Coppola, M. Children creating language: How nicaraguan sign language acquired a spatial grammar. *Psychological science*, 12(4):323–328, 2001.

Shi, B., Brentari, D., Shakhnarovich, G., and Livescu, K. Open-domain sign language translation learned from online video. *arXiv preprint arXiv:2205.12870*, 2022.

Shohieb, S. M., Elminir, H. K., and Riad, A. Signsworld atlas; a benchmark arabic sign language database. *Journal of King Saud University - Computer and Information Sciences*, 27(1):68–76, 2015. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2014.03.011. URL https://www.sciencedirect.com/science/article/pii/S1319157814000548.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Simon, T., Joo, H., Matthews, I., and Sheikh, Y. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1145–1153, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sincan, O. M. and Keles, H. Y. Autsl: A large scale multi-modal turkish sign language dataset and baseline methods. *IEEE Access*, 8:181340–181355, 2020. doi: 10.1109/ACCESS.2020.3028072.

Sites, M. Ieee standard for floating-point arithmetic. *IEEE computer society*, 2008.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. volume 15, pp. 1929–1958, 2014.

Starner, T. E. Visual recognition of american sign language using hidden markov models. Technical report, Massachusetts Inst Of Tech Cambridge Dept Of Brain And Cognitive Sciences, 1995.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Thangali, A., Nash, J. P., Sclaroff, S., and Neidle, C. Exploiting phonological constraints for handshape inference in asl video. In *CVPR 2011*, pp. 521–528. IEEE, 2011.

Tharwat, A., Gaber, T., Hassanien, A. E., Shahin, M. K., and Refaat, B. Sift-based arabic sign language recognition system. In Abraham, A., Krömer, P., and Snasel, V. (eds.), *Afro-European Conference for Industrial Advancement*, pp. 359–370, Cham, 2015. Springer International Publishing.

Tompson, J., Stein, M., Lecun, Y., and Perlin, K. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33 (5):169, 2014.

Tzirakis, P., Trigeorgis, G., Nicolaou, M. A., Schuller, B. W., and Zafeiriou, S. End-to-end multimodal emotion recognition using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1301–1309, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017a.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vogler, C. and Metaxas, D. Parallel hidden markov models for american sign language recognition. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pp. 116–122. IEEE, 1999.

Vogler, C. and Metaxas, D. Handshapes and movements: Multiple-channel asl recognition. j. carbonell, j. siekmann (eds.), gesture-based communication in human-computer interaction, lnai 2915, 2004.

Voskou, A., Panousis, K. P., Kosmopoulos, D., Metaxas, D. N., and Chatzis, S. Stochastic transformer networks with linear competing units: Application to end-to-end sl translation. In *Proc. ICCV*, 2021.

Voskou, A., Panousis, K. P., Partaourides, H., Tolias, K., and Chatzis, S. A new dataset for end-to-end sign language translation: The greek elementary school dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1966–1975, 2023.

Wainwright, M. J. and Jordan, M. I. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

Walawalkar, D. Grammatical facial expression recognition using customized deep neural network architecture. *arXiv preprint arXiv:1711.06303*, 2017.

Wang, H., Chai, X., Hong, X., Zhao, G., and Chen, X. Isolated sign language recognition with grassmann covariance matrices. *ACM Transactions on Accessible Computing (TACCESS)*, 8(4):14, 2016.

Wang, J., Liu, Z., Chorowski, J., Chen, Z., and Wu, Y. Robust 3d action recognition with random occupancy patterns. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C. (eds.), *Computer Vision – ECCV 2012*, pp. 872–885, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33709-3.

Wang, L., Hu, W., and Tan, T. Recent developments in human motion analysis. *Pattern recognition*, 36(3):585–601, 2003.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 568–578, 2021.

Wilson, A. D. and Bobick, A. F. Parametric hidden markov models for gesture recognition. *IEEE transactions on pattern analysis and machine intelligence*, 21(9):884–900, 1999.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 22–31, 2021.

Xiao, Q., Chang, X., Zhang, X., and Liu, X. Multi-information spatial–temporal lstm fusion continuous sign language neural machine translation. *Ieee Access*, 8:216718–216728, 2020.

Xu, Y. Implement long short-term memory recurrent neural network on grammatical facial expression recognition.

Yin, K. and Read, J. Better sign language translation with STMC-transformer. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 5975–5989, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

Zeiler, M. D. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

Zhou, H., Zhou, W., Zhou, Y., and Li, H. Spatial-temporal multi-cue network for continuous sign language recognition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 13009–13016. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/7001.

Zhou, H., Zhou, W., Qi, W., Pu, J., and Li, H. Improving sign language translation with monolingual data by sign back-translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1316–1325, 2021.

Zhu, G., Zhang, L., Mei, L., Shao, J., Song, J., and Shen, P. Large-scale isolated gesture recognition using pyramidal 3d convolutional networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 19–24. IEEE, 2016.