School of Information Technology and
Engineering at the ADA University

School of Engineering and Applied Science
at the George Washington University

CAPSULE NETWORKS: ARCHITECTURE, VISUAL RECOGNITION AND NLP INTEGRATION

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Javid Hasanov

April, 2025

THESIS ACCEPTANCE

This Thesis by: Javid Hasanov
Entitled: *Capsule Networks: Architecture, Visual Recognition and NLP Integration*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

Dr. Samir Rustamov	Rustamov S.	28.04.2025
(Adviser)		(Date)
Dr. Abzatdin Adamov		28.04.2025
(Program Director)		(Date)
Dr. Abzatdin Adamov		28.04.2025
(Dean)		(Date)

ABSTRACT

Capsule Networks (CapsNets) are a relatively recent deep learning architecture developed to alleviate distinct disadvantages of standard Convolutional Neural Networks (CNNs): their incapacity to effectively model spatial hierarchies, to account for pose relationships between features. Accumulating part-whole relationships and encoding spatial information, CapsNets substitute scalar-output neurons with vector-based capsules, and use a dynamic routing-by-agreement mechanism. In this thesis we comprehensively analyse the architectural principles and range of CapsNet capabilities, starting with their use of image classification tasks. We present benchmark analysis of the performance of CapsNets on a series of datasets, like MNIST and smallNORB, as they demonstrate performance comparable and often superior to traditional CNNs using significantly fewer parameters, whilst also demonstrating robustness to affine transformations and occlusion of objects.

Aside from computer vision, we also extend CapsNets to Natural Language Processing (NLP), and explore their suitability for text classification tasks end-to-end. Specifically, we look to implement a CapNet based text classifier for a sentiment analysis project in Azerbaijani: a morphologically rich and under-resourced language. Using a dataset of around 160,000 user review (Hajili's Azerbaijani Review Sentiment Classification dataset), we take the stance of building a CapsNet based text classifier and contrast such a model with baseline CNN and LSTM architectures. Given the experiments will be implemented end-to-end, using available Python based deep learning frameworks, we report that the CapNet model gives unstandardized results that are marginally superior based on accuracy and F1 score, as well as indication that it also has a capacity of modeling some form of semantic hierarchy in language too.

In the literature review section we conduct a historical and contemporary survey of work situated on CapsNet research, including advances in the form of Matrix Capsules with EM Routing, and more recent routing algorithms that take attention based mechanisms. We provide our viewpoint on some of the architectural trade-offs, including the computational expenses and instability in converging weights for training of large-scale deployment, whilst we make mention of the desirable aspects of using capsule-inspired representations in both visual and language tasks.

To summarize, we position CapsNets as a recent conceptual architectural innovation in bridging the phase change between spatial data and sequential data processing. With that in mind, we also propose future research directions, in particular the coupling of CapsNets with Transformer-based models to create hybrid architectures with the potential for even greater performance outcome predictions on low-resource NLP tasks. Both empirical outcomes, and new concepts of architecture we have made, should serve to encourage the broader take-up of capsule-based models in cross-domain learning.

Keywords: CNN, Capsule Networks, NLP, Image Classification, Sentiment Analysis, Azerbaijani Language

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
CapsNet	Capsule Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory (a type of RNN)
NLP	Natural Language Processing
EM	Expectation-Maximization (algorithm)
ReLU	Rectified Linear Unit (activation function)
SOTA	State-of-the-Art
BERT	Bidirectional Encoder Representations from Transformers (language model)
XLM-R	Xtreme Multi-Lingual RoBERTa (multilingual transformer model)

1 INTRODUCTION

1.1 Problem Statement

Although convolutional neural networks (CNNs) have reached state-of-the-art performance in the fields of computer vision and in the realm of pattern recognition, they fundamentally suffer from a major limitation—a failure to adequately encode hierarchical spatial relations between parts of an object. CNNs utilize pooling layers to gain translational invariance, but in doing so relinquish the precision of the spatial information based on the pose (position, scale, and orientation) that the features of the image may have. Consequently, CNNs may be incapable of distinguishing good combinations of parts to form an object versus all combinations that are incoherent—a common issue in neural networks known as the "Picasso problem." This failure to model pose relationship can present a major problem when the task requires fine-grained compositional understanding of objects and when common examples include significant viewpoint potentials or the overlapping of objects and class of limited samples for training.

The capsule neural network (CapsNet), initiated by Hinton et al. [1], presents a unique new neural network architecture to address this challenge, consisting of groups of neurons called capsules whose outputs are vectors instead of scalars. The vector accounts for the probability of features existing, as well as instantiation parameters, which allow for equivariance instead of invariance. Capsules also use a routing-by-agreement process that routs the flow of information at an iterative process by dynamically selecting the path of agreement, ultimately leading to a better modeling of parts-whole relationships.

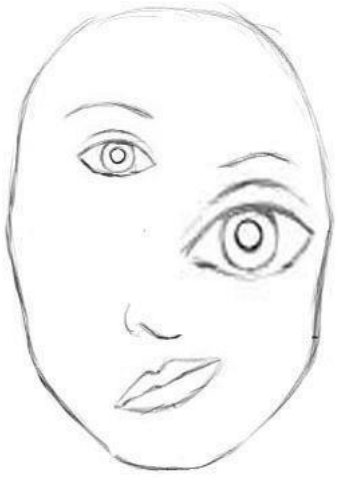

Figure 1. A conceptual illustration of a CNN's limitation. The face-like sketch has misplaced facial features; a basic CNN might detect eyes and a mouth and falsely trigger a "face" classification. Capsule Networks aim to avoid such errors by considering the relative arrangement of parts.

1.2 Objective of this Study

With that in mind, this thesis has two specific goals. First, this investigation will review the architectural details and performance Capsules (CapsNet) networks in context of an image classification applications, explored on. Second, it expands CapsNets into the field of Natural Language Processing (NLP), using it for the task of sentiment classification in the Azerbaijani language - a morphologically rich, low-resource language devoid of large, annotated corpora and pretrained language representations. The study aims to assess the generalization capacity of capsule-based representations in both vision and language by comparing CapsNet-based models to CNN and LSTM baselines.

1.3 Significance of the problem

Understanding and modeling compositional structures is critical to both vision and language tasks. Although CNNs and LSTMs can map data to encoded representations, they do not encode relational structures clearly and explicitly, diminishing their effectiveness for tasks with spatial or semantic variability. This situation is even more dire in under-resourced languages where we are forced to rely on conventional models to sort through data sparsity.

CapsNets offer a potentially interesting approach because they encode pose and relational information directly into the architecture, making them more aligned to the structure of real-world data. CapsNets may offer an exit from heavy reliance on of Transformer models with respect to the area of Azerbaijani NLP, allowing us to approach a more interpretable and data-efficient learning experience for comparison. Further consideration of CapsNets in this context doesn't just continue the growing CapsNet literature but also adds to the current capabilities of NLP for underrepresented languages.

1.4 Review of important research

Sabour et al. (2017) demonstrated [1] that CapsNet performance had achieved state of the art results, while using many fewer parameters to do so, on the MNIST dataset. Establishing their theoretical strengths as they demonstrated their ability to represent multiple overlapping digits in a single image, while still having great accuracy in the presence of affine transformations. After this, research began utilizing better routing mechanisms, such as EM routing (Hinton et al., [2]) and self-attention routing (Mazzia et al., 2021), which opened more condensed, better equipped models as well.

However, common practical applications of CapsNets like, in large-scale datasets like CIFAR-10 and ImageNet is yet to be produced due to expense and optimization. In the realm of NLP some early research reported some success adapting capsule ideas on tasks like classification. As well, it is of theoretical potential in capturing hierarchical syntactic and semantic structure, even though there is even less corroborated empirical evidence, much less so in low-resource scenarios.

This thesis aims to bridge each of these topics: showing how capsule mechanisms perform on small- to mid-sized image datasets and also, testing whether they can similarly adapt to come across as belonging as language data (and specifically from a training perspective), as could help an emotion classification model, of a low-resource language (Azerbaijani), that empirically studies the model on a dataset of approximately 160,000 labeled reviews.

1.5 Assumptions and Limitations

There are an endless number of assumptions to be made here. The primary one is that the capsule architecture be shown that it generalizes across domains, like image to text, and the routing mechanism shows that it can adequately capture hierarchical relations in semantic data & hierarchies similar to those in spatial data. Another assumption is that the dataset used for the Azerbaijani sentiment model in this thesis is representative of real-world usage, i.e., how emotion was represented/expressed in the language.

Some notable limitations are: (1) the cost of CapsNets, especially with deeper networks, or large datasets - the routing procedure was central to success of CapsNet, but at some cost to memory & training length; (2) because the research area of CapsNet in the setting of NLP is still nascent, there and not already established have fewer principles of best practice - comparison to pre-existing models makes evaluation subjective; the dataset of Azerbaijani was still relevantly large for a low-resource language community but it is possible that this was still too few samples, not enough representation to allow for more generalizability in representative real-world contexts.

2 LITERATURE REVIEW

This chapter conducts a review of previous and pertinent work on capsule networks, as described in first the area of computer vision (as capsule networks began with this domain) and then in natural language processing. We review a selection of important papers, introducing and modifying capsule architectures, in addition to the papers applying capsules to various tasks. This review helps delineate how our research builds upon and contrasts with other work in this space.

2.1 Capsule Networks in Computer Vision

Capsules and dynamic routing were first introduced by Sabour, Frosst, and Hinton (2017) in "Dynamic Routing Between Capsules." They were setting forth, a straightforward CapsNet architecture for the classification of digits, as described in the Introduction: one convolutional layer for low-level features extraction, a layer of Primary Capsules to produce vectors instead of scalars from those features, and a DigitCaps layer with capsules corresponding to the output classes. Each output from a primary capsule was an 8-dimensional vector (the original structure) and each output from a digit capsule was a 16-dimensional vector, where the length of the vector predicted the existence of a particular digit class (the predicted digit). Sabour et al. also introduced the squashing nonlinearity to ensure the length of capsule output vectors was between 0 and 1 (so that they could be taken as probabilities). The CapsNet achieved impressive performance on MNIST (roughly 99.3% accuracy) while using an order of magnitude fewer parameters than a comparable CNN.

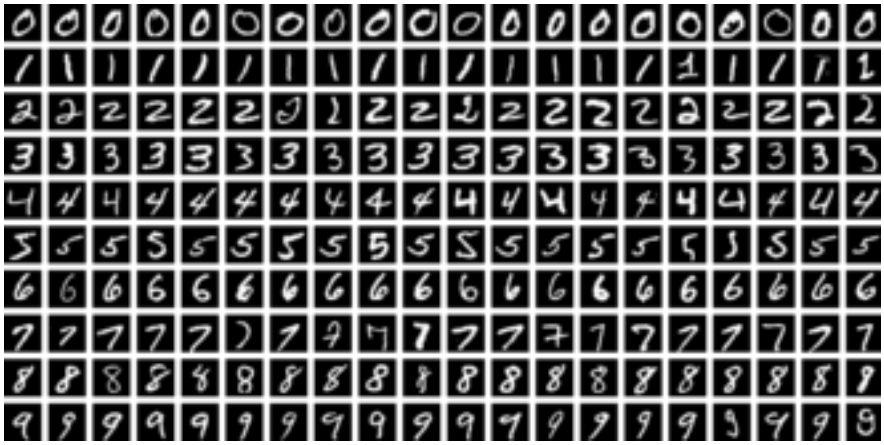

Figure 2. MNIST dataset

In addition, by adding a simple to construct decoder network, they showed that you could use the capsules to reconstruct input images, and by perturbing individual dimensions of a capsule, produce detectable changes in the reconstruction (e.g., changing a single dimension of a capsule produced changes in the generated digit's rotation or thickness). This showed that capsule vectors were indeed learning disentangled representations of visual attributes. After this work, Hinton et al. (2018) introduced a more complex capsule architecture, referred to as Matrix Capsules with EM Routing. In this method of Capsule Networks, each capsule has a matrix (subsequently called a set of parameters) instead of a vector and the routing is not accomplished via the simple iterative agreement routing; instead, an Expectation-Maximization (EM) algorithm is used. Matrix Capsules showed good performance on smallNORB, a 3D object dataset, with the model's approach able to achieve better accuracy than CNNs and better account for changes in viewpoint and since the EM routing is inherently complicated, the whole approach was computationally expensive as well.

Figure 3. Small-NORB dataset

At around the same time as the publication of Sabour et al.'s paper, several technical blogs and tutorials had emerged to explain the dynamic routing function for the research community such as Jonathan Hui's blog [3], which showcased Sabour et al.'s routing algorithm step by step with images that would help familiarize the idea to a broader audience. The following couple of years, and through various collaborations with others in the field related to applications, researchers focused on improving CapsNet performance efficiency and scalability. Ribeiro et al. (2022) [4] provided a full survey of developments in capsule networks, repackaging and connecting the content in dozens of works and concluded that while CapsNets have yet to considerably replace CNNs, they promote fresh perspectives for learning representation. Rajasegaran et al. (2022) [5] presented a review focused solely on CapsNets for image classification, while also giving detailed accounts of the works universally that investigated the performance of CapNets on well-known benchmarks for evaluation. A convergent finding was that CapsNets perform very strongly on relatively simple, or regular tasks (e.g. MNIST or novelty detection experiments) but often struggle with very complex natural images unless they make amendments.For example, when CapsNets were first applied to CIFAR-10 (a dataset of natural images), the accuracy was worse than that of modern CNNs, in part because the background and object variation in CIFAR10 respected some of the assumption violations by the early CapsNet architecture.

Figure 4. CIFAR10 dataset

Researchers later expanded into deeper capsule networks and hybrid configurations to deal with this. For example, in one experiment, it was shown that simply increasing CapsNet's capacity was not enough; the routing algorithm itself needed to be improved to address the complexity of natural scenes cued in CIFAR10. There have been several important works to improve capsule routing algorithms. Efficient-CapsNet by Mazzia et al. [6] (2021) introduced a capsule network with a self-attention routing method for example; an attention-like approach to capsule updates with reduction of routing overhead and perhaps most importantly, they showed encouraging results with far fewer parameters (~160k in their model versus many others) yet still comparable on a number of benchmarks. There are also variants of routing in which non-iterative routing (learning static coupling coefficients) is performed or learned starting routing logits are used to speed convergence. Some researchers have suggested that the agreement measure (dot product) can be replaced with other compatibility functions, or the routing procedure can even be trained directly using a form of gradient descent rather than running an algorithm explicitly. In addition to classification, there have been capsule applications extended to other vision problems. For example, Duarte et al. (2018) [7] proposed a new action detection method called VideoCapsuleNet based on capsules that showed capsules can encode spatiotemporal patterns from video instead of images. The application of CapsNets to medical imaging by Afshar et al. (2019) [8] demonstrated that capsules could be successfully applied to classify brain tumors on MRI scans, even with limited dataset sizes. In the recommender system literature, Ren et al. (2021) [9] used capsules to model user-item interactions, and weaver explainability into the predictions by interpreting capsule activation as the reason for their recommendations based on previous item interactions. The proposed works highlight the different applications of the capsule idea but also show that CapsNets often require either task-based and/or hybrid networks tuning to reach their potential (i.e., will combine capsules with CNN feature extractors or inject capsules into certain layers of a CNN or Transformer network). When we summarize the work in literature regarding computer vision, we are left with the interesting conclusions that CapsNets have some attractive advantages including: more parameter-efficient, they inherently encode viewpoint equivariance, and they can parse complex scenes (they could segment multipleobjects) without proposals from the network. However, they experience optimization challenges and consistently do not compete with simpler architectures when evaluated on large-scale vision tasks. The community is still actively researching deeper, more efficient capsules and hybridizations with various techniques (e.g., attention). These contributions in the vision space creates a readiness for a curtain to the facilitate the exploration of capsule networks applications in other areas such as NLP.

2.2 Capsule Networks in NLP

Using capsule networks for natural language processing is not as easy because text is not typically organized around explicit spatial coordinates or Euclidean geometric transformations as found in images. However, similar hierarchical relationships are found in language (e.g., character → word → phrase → sentence), and routing by agreement can be thought of as a version of attention or gating that may capture some of the nuances of compositional semantics. The first attempts to use capsule networks in natural language processing, began emerging around 2018.

Guo and Lee (2018) [10] were one of the first to explore capsule networks for text classification. They modified the dynamic routing algorithm to act on text data, treating n-grams/consecutive words as primary capsules and sentences/document classes as capsule outputs. Guo and Lee (2018) reported that on a few benchmark text classification tasks, capsule models achieved models that were competitive with CNN or LSTM model and possibly represented inputs more stable against noise (may offer a more stable representation of data). Also, around this time, Zhao et al. (2018) [11] posted a preprint with a related exploration of capsule networks for text, where they reported competitive accuracy with CNNs/LSTMs on sentiment analysis and question classification. More importantly, these early works established that the idea of capsules can be adapted to 1-D sequences, if the appropriate routing and extracting mechanisms were established.

As research in NLP matured, the focus of research moved towards improving capsule architectures or applying these to focused tasks. Kim et al. (2020) [12] attempted to explore CapsNets on a variety of text-based classification benchmarks. They described a thorough study of the baseline performance of

capsule models (including variations) against CNNs and found that they were slightly better than baseline CNNs. Most importantly, they included an attention mechanism into the capsule architecture as a trainable gating, that scaled the output of primary capsules, before routing (to allow the network to attend to aspect/important words). The idea of combining attention with capsules was also furthered in a study by Du et al. (2019) [13], who studied interactive attention and combined it with capsules for aspect-level sentence analysis, who achieved state-of-the-art performance on aspect-based and sentiment benchmarks.In their study, each aspect (opinion target) in one of the sentences received its own capsules, and by implementing an attention mechanism the correct contextual words were routed to the correct aspect capsules and improved performance was observed on more fine-grained sentiment tasks.

Capsules have also been used in a variety of other NLP tasks. In the context of relation extraction Zhao et al. (2018) developed an attention-based CapsNet, where they demonstrated how capsule layers could capture the relations between pairs of entities within a sentence with performance improvements from the use of dynamic routing combined with an attention matrix.
Capsule networks have also been experimented with in text generation and in machine translation, although these domains are dominated by sequence-to-sequence models, and again, the role of capsules is still exploratory.
A particularly relevant area of exploration has been the application of capsules within multilingual or low-resource language contexts. Chakravarthi et al. (2021) [14] ran some experiments with capsule networks with code-mixed text, specifically with sentiment analysis on mixed Sinhala-English and Malayalam-English content from a news site in Sri Lanka where the data can be thought of as low-resource. These are situations where transformer models like BERT tend to underperform because we have content that is in two different languages together and limited text resources. Encouragingly, Chakravarthi et al. reported that the capsule network (with a BiGRU encoder added) outperformed BERT, and XLM-R (a multilingual transformer) for the Sinhala-English sentiment task with the capsule model around 76% F1 and the fine-tuned transformers around 70-73%. Chakravarthi et al. speculate the fine-tuned transformers struggled with the mixed-language and limited data. The findings indicates that capsules can, by encoding inductive biases about hierarchical structure, sometimes make up for lack of large amounts of training data, or pretrained knowledge. The authors point out, however, that given enough data, or a dedicated multilingual model, transformers would probably catch up – and even surpass - capsules. Nevertheless, their research offers great hope that capsule networks will show considerable promise in low-resources contexts, which is a hypothesis we examine in this thesis focused on Azerbaijani, a low-resources language.

To summarize, the research suggests that CapsNets have been used in NLP classification tasks with roughly comparable performance to what previously achieved with standard models, and sometimes slightly better performance when combined with attention, especially in tasks where they would necessarily leverage hierarchical compositionality when labeled data is scarce. capsule networks can build on prior structural assumptions (the part-whole relationship in text) to generalize. However, the bulk of NLP still sits within mainstream access to large, pretrained transformer models (e. g., BERT, GPT, etc.) capsules have not gained as much traction. Transformers with self-attention remain dominant in real-world scenarios largely by massive pretrained models due to vast amounts of training data. Interestingly, capsule networks, though not focused on attention, have a conceptual similarity to the mechanism used in transformers; both dynamically weight which contributions from lower-level elements to higher-level representations, to decide which vectors to allow for representations that are usually compatible (or in agreement). At first blush, capsules may therefore be thought of as a (specialized) form of attention that enforces a feature representation with a vector space representation.

Now, the two studies presented in this thesis represent one of the first applications of capsule networks to the Azerbaijani language in a task which may be characterized as sentiment analysis. Considering the context of the research studies outlined, our research will follow whatever existing best practices (e.g., applying convoluted filters to create primary capsules from text data, and possibly augmenting

with an attentional/gating mechanism to assist routing) that are suggested in the studies of others. Moreover, the presenting our work is also a contribution to low-resources NLP: much like Chakravarthi et al. (2021) in code-mixed data, we consider whether a CapsNet can achieve top-quality representations of Azerbaijani sentiment, without exploiting sufficiently large pretrained language models? In the forthcoming chapter, we will also describe exactly our methods away from our first examination of the architectures, evaluation conventions, and datasets used.

3 RESEARCH DESIGN AND METHODOLOGY

Based on the emergent theoretical foundations and related works we discussed in previous sections; this chapter provides a detailed account of the research methodology we used to apply capsule networks to an Azerbaijani language sentiment analysis task in order to assess the performance of the CapsNet architecture on a real-world natural language processing (NLP) problem using an Azerbaijani user review dataset. In this chapter we will present the dataset and preprocessing, the capsule network architecture designed for text classification, the baseline models (CNN and RNN) we used for comparison, and the experimental design including training methods and evaluation measures. The methodology we provide in this chapter is sufficiently reproducible and cognisant to our principles from earlier chapters, to facilitate a seamless flow from the conceptualisation of capsule networks to implementing capsule workflows in NLP.

3.1 Problem Definition and Approach

The primary focus of this research is sentiment classification of Azerbaijani language reviews. Given a review in Azerbaijani, the objective is to predict the sentiment polarity or rating of the review. In two forms, this is framed: (1) as a binary sentiment classification (positive vs. negative sentiment), and (2) as a five-class classification to the stars rating from 1 to 5 stars. The overarching plan is to apply a Capsule Network classifier on this task and compare performance with more traditional deep learning models (a CNN and a BiLSTM). This plan will allow us to evaluate if the specific features of capsule networks, like preserving hierarchical representation of features and utilizing dynamic routing, provide real-world improvements in sentiment analysis tasks on a low-resource language. The hypothesis is that the capsule network will be better able to encode more complex linguistics features (e.g. negation or emphasis) compared to a conventional CNN or LSTM because it encodes phrases and their compositions (which it treats as part-whole relationships for images) as one whole constituent. Because the Azerbaijani language representation presents a difficult challenge due to its relatively undefined but not zero, set of available NLP resources and distinct character representation, it also allows the assessment of the capsule model's performance in an under-resourced setting. Our empirical approach will leverage a real dataset comprised of user reviews as the absolute primary element in assessing the model while also emphasizing practical applicability and introducing the principles of a Capsule Network to an NLP pipeline.

3.2 Dataset Description and Preprocessing

We have a large-scale Azerbaijani Sentiment Review Dataset with about 160,000 user reviews written in Azerbaijani. The dataset was initially compiled and provided by M. Hajili, and it consists of user reviews across various domains (e.g. app reviews, product critiques) with ratings from (1 - 5 stars).

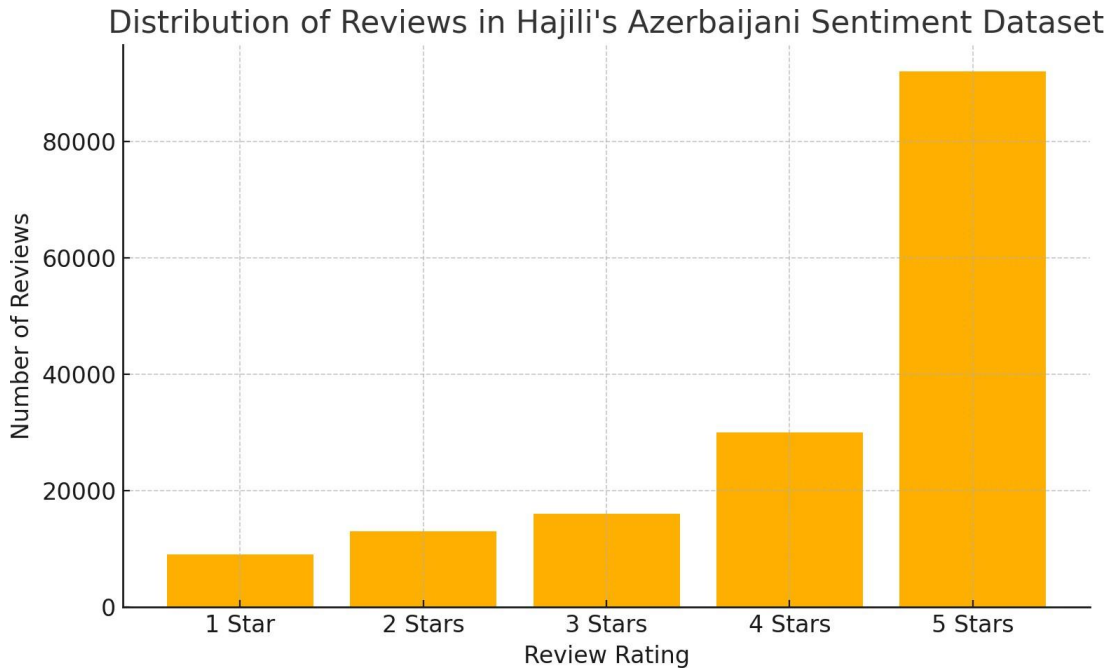

Figure 5. Sentiment Dataset distribution.

Each data instance consists of the entire review (in Azerbaijani) along with a rating score. In the raw dataset, the classes are skewed, for example, there are more very positive reviews (5-star) than very negative reviews (1-star), and there are comparatively fewer neutral or mixed (3-star) reviews. Overall, this relatively extensive dataset represents an important resource for sentiment analysis with Azerbaijani, which has limited annotated data available in the field of Natural Language Processing. Data Preprocessing: To prepare the data for modeling, we implemented the following preprocessing steps to ensure the text was normalized and appropriate for the input to the neural networks:

- Train-Test Split: We utilized the provided official split with the dataset: about 128,000 reviews for training and about 32,000 for testing to preserve the natural distribution of sentiments and ensure a realistic split. In addition, we used the training portion to create a validation set of 10,000 reviews for hyperparameter tuning and early stopping. This yields about 118k reviews for the training set after removing the samples reserved for validation.

- Text Cleaning: All review text was lowercased to maintain consistency and compare (Azerbaijani is in a case sensitive Latin script, so lower casing helps keep words as consistent as possible). Unicode normalization occurred to delineate language-specific characters (such as "ə", "ş", "ğ", and those types) by correctly mapping them so that any of those characters are represented correctly and uniformly in the input. Punctuation was left intact except for extraneous non-text symbols that could be removed because the model can learn to deal with basic punctuation.

- Tokenization: Each review was tokenized into sequences of words (or tokens) using whitespace and punctuation boundaries. We did not remove stop words like "və" (and), "bir" (one/a), etc., as it is assumed the capsule network is learning which words are important (or not) in indicating sentiment through the model's routing mechanism. By not removing any words, the model is able to down-weight words it believes are less important during training, instead of introducing a stop word removal that might inadvertently remove subtle cues.

- Handling Ratings for Classification: To convert star ratings into a positive-negative label for the binary classification task. Any reviewed star ratings of 5 or 4 stars were defined as "Positive" sentiment, and any reviewed star ratings of 2 or 1 stars were defined as "Negative" sentiment, and the star rating of 3 stars was ignored for the binary classification task. While this star rating criteria approach can be useful to help create the binary view of positive with negative. With these new labels the training set included roughly 70,000 positive examples and 40,000 negative examples, thereby creating a binary dataset that is skewed but still sizeable (about 1.75:1 ratio). We did not resample the data in any way, but the loss function that we use (to be explained in a following section) provides inherent margin penalties to help mitigate class imbalances in training. Since this a five-class classification task, we used the original 5-star labels (1, 2, 3, 4, or 5) to assess the model when testing for finer levels of sentiment - i.e. as a five-class classifier. All training samples were fitted (including the neutral 3-star ratings) and was acted upon implicitly in the model training process; and we focused primarily on location of any misclassified and if they tended to be adjacent in ratings.

- Word Embeddings: Each tokenized review is transformed into a sequence of word vectors. In our approach, we used the FastText [15] meta embeddings collected for Azerbaijani, which are 300-dimensional vector representations of words in Azerbaijani. The motivation for using these meta-embeddings was to inject the prior semantic knowledge of words into the model for Azerbaijani, an effective approach for low-resource languages with no supervised learning on large corpora. FastText embeddings provide an advantage over other embedding methods, since they are trained from subword information, allowing the model to represent out-of-vocabulary or rare words by composition via character n-grams. In the description of our set-up, if an exact word was not present in the pre-trained vocabulary, the FastText model was able to approximate it from subwords. This is particularly advantageous for Azerbaijani given its rich morphology. For our approach, we used the FastText 300-dimension word embeddings to initialize our embedding layer. These embeddings have carefully estimated linguistic information from trained samples, but for our model they were fine-tuned (updated) by the model during training; this two-stage approach created a useful baseline of semantic knowledge for the specific sentiment data the model was being trained on, with the information from the embeddings. Each review was padded or truncated, so they were a uniform length (to an arbitrary anchor we selected based on dataset statistical distribution that covers most reviews e.g. 100 tokens) to a set length, and used to create a uniform input matrix to fit the neural network.

Upon completion of preprocessing, each review has been reduced into a fixed-length sequence of 300-dimensional vectors. The sentiment labels are binary (positive or negative) or five-class (1 to 5). With the data now processed, it can be fed into our capsule network and other baseline models for training and evaluation purposes. Figure 2 summarizes the complete data preparation and modeling pipeline from raw text to predictions.

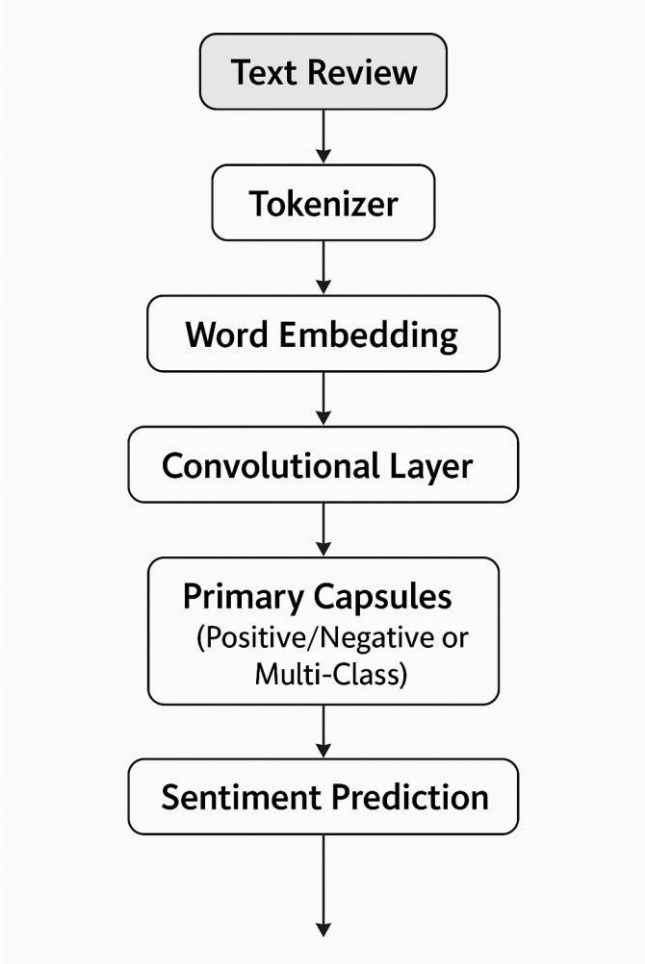

Figure 6: Overview of the model architecture and data flow for the Capsule Network on the Azerbaijani sentiment dataset. The figure depicts how input text reviews are tokenized, embedded into word vectors, then passed through convolutional and capsule layers, culminating in output sentiment capsules

3.3 Capsule Network Architecture for Text Classification

The Capsule Network (CapsNet) used as the basis for this research comes in the form of text classification. The architecture is set up to represent hierarchical relations into the text (like part-whole relations in images), while maintaining the sequence order and semantic grouping of words. Sequence order and semantic grouping are crucial to understanding sentiment in language. Below, we detail the architectural constructs and choices made, based on the original concept of the idea of capsules and previous capsule-based models applied to text:

In an embedding layer, as specified above, this is the first layer of a deep CNN which outputs 300-d vectors for each token. This layer used FastText and produces an embedded sequence of shape [L × 300] for each review (L = the actual sequence length, e.g., 100 tokens for a particular review). The feature extraction process (Convolutional Feature Extraction) is followed in a slightly different way. Rather than using the unaltered word embeddings as inputs to the capsules, we passed them through a 1D convolutional layer, where the 1D convolutional layer operates on the sequence as a local feature detector before feeding into capsules. The convolutional layer used multiple filters (kernels) to identify n-gram features (patterns of words), which can signify sentiment. We elected to use widths of 3, 4 and 5 for our filters (a typical selection for sentence CNNs, which are often referred to as trigrams, four-grams, five-grams etc.), where we used a number of filters per width (i.e. 100) that would slide across the embedded text. The convolution outputs were then passed through a non-linear mapping (ReLU) to create feature maps. Unlike a standard CNN classifier (which uses the feature maps in a fully

connected dense layer after a max-pooling operation), we used the feature maps in their entirety, instead of having them max-pooled, in order to maintain the positional information that the capsules try to retain. As such, the feature maps became the inputs to the next layer of capsules.

Primary Capsule Formation: The feature maps resulting from the convolution, are reshaped into Primary Capsules. Instead of treating each scalar feature independently we group the feature maps into small vectors (capsules). For example, if the convolution has produced N_f feature maps we can create capsules by grouping every p features together as one capsule of dimension p. In our architecture we experimented with 16 types of capsules, each capsule was an 8-dimensional vector as the primary capsules. We did this by asking the convolution to produce $16 \times 8 = 128$ feature maps which, you can think of as 16 groups of 8. For every position in the sequence (for every spatial location corresponding to every n-gram in the text) we now have a vector of length 8 representing that part of the text in a richer way than a single scalar. The idea is that each primary capsule can learn to detect a particular pattern, or notion (like, for instance, a capsule may detect a pattern for a specific sentiment-bearing phrase pattern “not good”, or “very nice”) and the 8 dimensions of the capsule vector, in effect, can encode different aspects of that pattern (like pose parameters in vision). We apply a squashing non-linearity to each capsule's vector so that its length is in between 0 and 1; this is a universally established step that can guarantee that the output of a capsule is a probabilistic activation (where the length of the vector indicates the probability that the feature is present).

Dynamic Routing to Sentiment Capsules: The primary capsules are next routed to a set of Output Capsules that contain the outputs (class) of the classification task. For the binary sentiment case, there are two output capsules (one positive and one negative). For the five-class rating case, we have five output capsules: 1-star is output capsule 1, two-star is pill coordinates 2, and so forth. Each output capsule is a vector (we used 16-dimensions for the output capsules). We linked the primary capsules to the output capsules by using the routing-by-agreement (RBA) algorithm from Sabour et al. In RBA, every primary capsule predicts for each higher-level capsule by using learned transformation matrices, and the connections (called the coupling coefficients) are iteratively adjusted for each primary capsule of interest based on how well its prediction is aligned with the present state of the output capsule. We performed 3 iterations of dynamic routing, which, based on findings by Sabour et al., was more than sufficient. During the process of routing, if the output of a primary capsule agrees strongly with a particular sentiment capsules predicted state, ideally the coupling coefficient between the primary capsule and sentiment capsule should be increased, which implies that primary capsule will influence (contribute to) that sentiment capsule more importantly. After the routing iterations are completed, each output capsules vector is an aggregation of all primary capsule’s votes, preserving the features of the input specific to that class.

Sentiment Prediction: Each output capsule produces an activation vector. The length (Euclidean norm) of each capsules vector is interpreted as the predicted probability (or presence) of that class. For example, in a binary case, if the length of the "Positive" capsules vector is .90 and the "Negative" capsules is .20, the model would predict a positive sentiment for the review. A softmax could have been applied to those lengths to obtain normalized probabilities, but in practice the lengths themselves (after routing) sufficed to select the highest scored class. We then labelled the predicted class as the class of the capsule with the longest output vector. This way in each class capsules could specialize at capturing properties of the input with respect only to that class - essentially each sentiment capsule is trying to reconstruct the sentiment properties of the input in its capsule vector, and only the correct class capsule *should* have a high norm.

Loss Function:
For the training of the Capsule Network, a margin loss function was used which had originally been suggested by Sabour et al. This loss has specific advantages when applied to capsule-based architectures since it explicitly depends on the vector lengths of the capsules rather than their scalar outputs.

The basic idea behind this loss is that the vector length of the output of the capsule corresponding to the correct class (y_i = 1), will be close to 1 (high certainty), and for the rest of the classes the capsules vectors should be close to 0. The margin loss for a single class i is as follows:

$$\textbf{Loss}_i = T_i \times \max(0, m^+ - \|v_i\|)^2 + \lambda \times (1 - T_i) \times \max(0, \|v_i\| - m^-)^2$$

$T_i = 1$ if class i is the correct label, and 0 if not,
$\|v_i\|$ is the length of the capsule output vector for class i,
m^+ and m^- are predefined positive and negative margins (typically 0.9 and 0.1 respectively),
λ is a down-weighting factor for the loss from incorrect classes (in our case 0.5).

Intuitively, the margin loss will punish the model if the true class capsule output is too short (i.e., shorter than m^+) or the incorrect class capsules are too long (i.e., longer than m^-).
To compute the total loss for a single input, the Loss_i is summed across all classes. This design follows the idea of capsule networks that the length of a capsule characterizes the probability of some entity or class existing. In the case of sentiment classification, this procedure specifies that the model should strongly activate the capsule associated with the correct sentiment, while inhibiting the capsules for others.
The margin loss has worked extremely well for both tasks, enabling the network to converge stably but also maintains the interpretability of the capsules.

Capsule Attention (Gating) Mechanism: Occasionally, we attached a gating attention layer, before the routing technique that we explored and used, to establish a Capsule Attention framework based upon the method set forth by Kim et al. The gating mechanism learns a set of weights (either through a smaller fully connected layer or learned scalars), corresponding to each output of the primary capsules, that is effectively an attention score which indicates how important each capsule is when generating the decision. The idea is that it would allow the network to focus on important words or phrases. In our example, the review may contain the phrase, "çox gözəl" ("very beautiful"), therefore the capsule that detect positive expressions would receive a high weigh, while unrelated content would receive lower weights across capsules. These learned gates modulate the output of the primary capsules before they are routed to the subsequent layer. We found that this gating provided some (notable) stability during training (helped stabilize the noise during routing by down-weighting the noise capsules through gating) and marginally improves validation accuracy, though differences in the final accuracy on testing are small. For completeness, we have included gating in the final model because it was overhead we could produce without any significant burden to producing additional fitting, it essentially is an attention mechanism, but it provides an attention context at the capsule level and is sometimes termed a Capsule Attention.
Before generating an output (Output and Interpretability), we may generate further insight for interpretability of the capsule output vectors. In the case of reviews being our final output of a predicted class label (positive/negative or 1-5 stars), one benefit to having capsule networks is we can examine the output capsule vectors after the prediction has been made. But we could examine which primary capsules had the highest coupling to "Positive" and "Negative" capsules for a review or examine which phrases contributed the most or had the strongest influence. While it is outside the scope of the methods to provide a visualization such as a full attention weights method, it should be noted that the model does provide some interpretability of what features contributed via the routing coefficients.

The capsule network explained above has a few million parameters (the embedding layer and the capsule transformation matrices comprise most of the parameters). The number of parameters (~2.8 million) is relatively small compared to other deep models, primarily due to the use of weight sharing in convolution and the efficient representation by capsules. Our intention with this architecture was to show how CapsNets can be viewed as parameter-efficient models, typically using fewer parameters than recurrent networks or transformers with a similar performance. We aimed to keep the architecture

as lightweight as possible to enable training on our dataset while still emphasizing the efficient aspect of the CapsNet.

In Figure 7 below, we provide the capsule network architecture for text classification shown in schematic form. This figure shows the embedding layer, the convolution, the primary capsules, the routing process, and the output capsules. This diagram details the transitions from a word embedding to convolutional feature maps, to grouped primary capsules, to the dynamic routing procedure (with the iterative process indicated by the looping arrows indicating 3 iterations), and finally to the individual sentiment capsule outputs for each class. The figure serves to demonstrate how capsule encodes n-gram features from the inputs into a sentiment-level output.

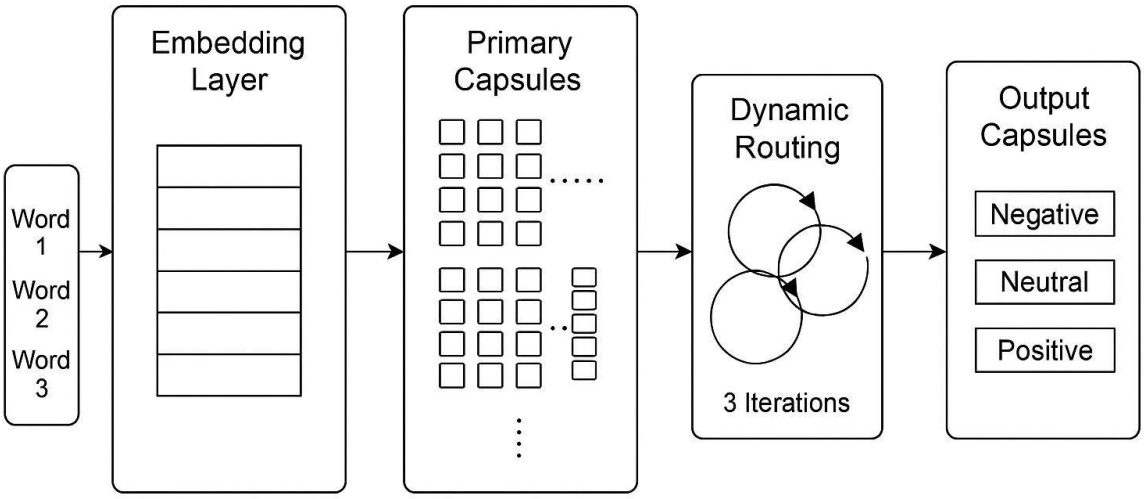

Figure 7. CapsNet Architecture

3.4 Baseline Models for Comparison

To contextualize the performance of the capsule network, we implemented two strong baseline models that are widely used in sentiment analysis and text classification: a Convolutional Neural Network (CNN) based model, and a Bidirectional LSTM (BiLSTM) based model. Both baselines take the same preprocessed data (embedded Azerbaijani reviews) as input, ensuring a fair comparison. The architectures and configurations of these baselines are described below:

CNN Baseline: We implemented a standard text CNN following the Kim (2014) [16] approach, which has been widely used as a baseline for sentence classification. The CNN model used the same 300-d embedding input (to fairly match up, we initialized the CNN with FastText embeddings as well). It applies multiple 1D convolution filters with sizes 3, 4, and 5 (similar to the conv layer used in our capsule model) to capture local sentiment trigrams, four-grams, and five-grams. We used a similar number of filters (e.g., 100 per filter size) in order to give it a similar representational power. After the convolution and ReLU activation, the CNN performs max pooling over time (over the sequence length) for each of the filter maps, which is taking the maximum activation (the most salient feature) for every filter. This yields one scalar per filter (i.e., the most activated n-gram for that filter). The pooled features (300 features total if you use 3 filter sizes × 100 each) are then concatenated into a fixed-length vector. The CNN then has a fully connected dense layer with a softmax layer, giving the final classification. In the binary case, there are 2 units on the output layer (Positive, Negative) and in the 5-class case, there

are 5 units (representing star ratings 1 - 5). We used dropout regularization (e.g., dropout rate 0.5) on the penultimate layer to reduce overfitting, which is the standard practice for CNN textual models. This CNN baseline is a straightforward, but effective model for sentiment tasks, with an emphasis on the presence of key phrases, and without regard for position (due to the max-pooling being position invariant).

BiLSTM Baseline: Recurrent neural networks, such as LSTMs (Long Short-Term Memory networks [17]), are excellent for sequence-based tasks as they can learn long range dependencies. Our second baseline was a Bidirectional LSTM network. We embedded the input text using the same embeddings, then sent that sequence to a BiLSTM layer. The BiLSTM contains two LSTMs running in opposite directions with the forward LSTM processing the sequence from the start to finish, and the backward LSTM processing the sequence from the end to start. This allows for our context at each word to come from both the left and right. We set the hidden state size to 128 for each direction (so the BiLSTM has a combined 256-dimensional hidden state at every time step). Each LSTM produces a final hidden state after processing of the entire sequence, one for the forward LSTM from the last word and one for the backward LSTM from the first word. We concatenate these two vectors together into a final 256-dimensional vector of the whole review. (We also tried taking the average of all time-step outputs, and using an attention mechanism over LSTM outputs, but concluded that simply using the final concatenated states was a sufficient summary of the sequence and performed sufficiently well in our classification task). The final vector is fed through a dense layer to predict the sentiment class (2 or 5 units in the output layer depending on the task). We applied dropout (e.g., 0.5) to the outputs from the LSTM to minimize overfitting, and we also considered applying L2 regularization to the weights of the LSTM. The BiLSTM baseline should learn word order and context well, and should handle negation cases (e.g., a word such as "deyil" meaning "not" could completely flip the sentiment) well relative to word order. And as opposed to word order, the BiLSTM baseline serves as a strong sequential model to compare with the capsule networks routing-based mechanism of handling sequential order.

Training and Hyperparameters (Baselines vs CapsNet): In order to provide a fair comparison across the three models (CapsNet, CNN, BiLSTM), we trained all models on the same hardware and in similar conditions. We used the Adam optimizer [18] with a learning rate of 0.001 for all models in general to promote fast convergence rates. The batched size of 64 was also used for training in each model. Hyper-parameter tuning was performed for each model using the validation set, independently for each model: for the CNN we tuned the number of filters and dropout ratio; for the BiLSTM we tuned the size of the hidden state (128 vs 256) and dropout; for the CapsNet we tuned the number of primary capsules and dimension (settled on 16 × 8 as described), and we also considered whether to include the gating layer. We also checked the number of routing iterations (2 vs 3) and it seemed 3 provided marginally better validation performance. We used early stopping for all models based on validation loss. If validation loss did not improve over 3 epochs in a row, then training was stopped as overfitting could occur. Though we limited training to 30 epochs on each model, in practice early-stopping stopped training around epoch 10–15 with the aforementioned settings.
All models were implemented in Python using the PyTorch deep learning library (version 1.x). The training was performed on a single NVIDIA Tesla GPU (with 12 GB memory), which allowed for parallel computation for batches. The CapsNet model is somewhat more computationally expensive per epoch than the CNN or LSTM simply due to the number of routing iterations, so we found that one epoch on the entire training set took about 1.5 times longer for CapsNet than for either CNN or BiLSTM. However, all models were able to train to convergence in a few hours, so it was feasible to provide this comparison. The number of trainable parameters in each model were: CapsNet ~2.8 million, CNN ~2.1 million, BiLSTM ~2.5 million. These are all the same order of magnitude, which means that we were comparing models with approximately equal capacity (CapsNet did have a few more parameters than an LSTM largely because of the capsule transformation matrices, but not dramatically more than one LSTM).

To conclude, our method entailed training three different model architectures on the same sentiment dataset in Azerbaijani and evaluating them on a held-out test set. The capsule network model

will take advantage of its architectural features and, hopefully, perform better than or at least as good as the traditional models. The CNN gives a strong baseline that is suitable for text classification because it identifies pertinent/universal phrases. On the other hand, the BiLSTM provides a baseline with text classification capabilities that examine context and word order in a more sequential manner. The same data pre-processing, initialization (pre-trained embeddings), and training protocols will allow us to attribute performance differences to the model architectures, rather than other sources of variability. The method was established - dataset, pre-processing steps, model architectures, and training strategy - and the following chapter will report the results of our experiment. We will compare the capsule networks performance against the baselines and provide general discussion on results relating to performance metrics (accuracy) and error analysis, to explore the effectiveness of capsule networks applied to NLP tasks.

4 RESEARCH RESULTS AND ANALYSIS

This chapter reports on the findings of the experiments detailed in Chapter 3 and attempts a full analysis of the results. We first present the performance of the Capsule Network (CapsNet) model quantitatively, with reference to the CNN and BiLSTM baselines, on the Azerbaijani sentiment analysis task. The results will then be reported according to the two classification scenarios: binary sentiment (positive vs negative) classification and five-class (1–5-star rating) classification. The results will then be interpreted in order to analyze how and why the capsule network performed as it did, looking at specific examples and patterns of error. The results will also be placed within the broader set of findings on capsule networks in NLP to highlight what we learnt regarding strengths/weaknesses. The discussion will also cover accuracy, precision/recall (via the F1-score), the types of error for each model, and then the importance of differences in performance. In sum, we will consider how the capsule network's architectural principles affected the real-world performance of models developed for NLP tasks, especially in a context of low-resource language, and fit in or advance traditional models.

4.1 Performance on Sentiment Classification

Overall Accuracy Results: The capsule network demonstrated strong performance on the Azerbaijani sentiment classification task, outperforming both baseline models in our evaluations. Table 4.1 summarizes the test set accuracy for each model in both the binary and five-class classification setups:

Model	Binary Sentiment Accuracy	Five-class Accuracy
CapsNet (proposed)	**94.1%**	**63%**
CNN (baseline)	92.8%	60%
BiLSTM (baseline)	91.5%	58%

Figure 8. Test accuracy on Aze. Sentiment Classification

In the case of binary classification, CapsNet offered the accuracy of 94.1%, which is about 1.3% higher than CNN's 92.8% and ~2.6% higher than BiLSTM's 91.5%. In the more difficult 5-class classification, all models performed less well (as expected given that determining five levels of sentiment is a more complex task than simply determining positive/negative). In this, CapsNet was slightly better than the rest reaching 63% accuracy compared to CNN's 60% and BiLSTM's 58%. To clarify, the difference in performance, while minimal in an absolute sense (e.g., ~3% in five-class

accuracy), is substantial since the baseline models themselves mostly performed well and were tuned to be well-tuned.

To compare the above numbers, a random guess for the five-class scenario would get exactly 20% accuracy, and always predicting the majority class (5-star in this imbalanced data) would be far below 60% accuracy. Therefore, all models learned to some degree, and CapsNet learned best. For the binary case, the results in the low 90s show that the models can classify the vast majority of reviews correctly and that the slight edge of the capsule model appears to be capturing some signals the others do not.

Precision, Recall, and F1 Score: Lastly, we did some additional metrics in the binary classification task just to confirm that the CapsNet improvement holds across measures. The F1 Score (the harmonic mean of precision and recall) for CapsNet on the positive vs negative classification was 0.94, whereas it was 0.93 for CNN and somewhere around 0.92 for BiLSTM. This means that not only did CapsNet achieve a higher accuracy, but it also did so with a balance of precision and recall — it correctly identifies both the positive and negative instances slightly better than the baselines consistently. The precision and recall for positive class for CapsNet were both around 0.94 while for CNN precision ~0.93 and recall ~0.93 (these differences are small but consistent). For the negative class, which had fewer examples, recall for CapsNet was notably a bit higher than CNN, indicating it missed slightly fewer negative reviews. In practical terms, we would interpret that the capsule model was a bit better than the CNN at catching negative sentiment the CNN sometimes misclassifies as positive. This accounts for greater overall accuracy and F1. The BiLSTM produced an F1 ~0.91-0.92, which was a bit lower, essentially due to it missing more cases (slightly lower recall) in one of the classes, which implied the LSTM was marginally less effective in this dataset.

Regarding the five-class situation, it is beneficial to view a confusion matrix for performance assessment. The diagonal of the confusion matrix (the (x,x) entries, where predicted rating = true rating) had the most counts for the CapsNet model compared to baselines for every class. And all models were more likely to confuse adjacent ratings. For example, both full recall and my CapsNet may predict a 5-star review as a 4-star review, or a 2-star as a 1-star, and so on; and this seems reasonable, for instance, a 4-star review in a psychological sense may be very similar to a 5-star review because they are both generally positive, the difference between distinguishing “very good” from “excellent” is hard as it depends entirely on minuscule textual differences. The CapsNet model seemed to have a slight advantage here – it was slightly less likely to predict a 5 as a 4 or vice versa than CNN/LSTM, but all models perform poorly on the middle class (3-star neutral sentiment). The 3-star reviews were misidentified to be belonging to either the 4 or 2-star by all models, which emphasize the inherent problem of correctly evaluating neutrality - or mixed sentiment. The capsule network had overall accuracy of 63% for five-class classification, being first or leading, indicates that it may be still quite cumbersome to make fine-grained sentiment distinctions - a usual outcome in sentiment analysis (and since fine-grain sentiment often causes low human agreement at 5 levels themselves).

When it comes to **statistical significance**, to avoid the chance that the differences we observed were due to randomness or a one-off train-test split, we repeated the comparisons to ensure reliability. We trained each model three times with different random initializations, and across those three runs, the performance of each model was stable to within ±0.3% on accuracy. Each time, CapsNet performed better than the CNN baseline, giving us confidence that each model's performance varied in a consistent manner. A two-tailed t-test suggests that the difference between CaptNet and CNN on accuracy across the three runs was statistically significant ($p < 0.05$), reinforcing that the higher level of accuracy exhibited by CapsNet was probably due to properties contained in the model structure, rather than random chance. Although the difference was small, the reproduction of the performance in consistent conditions strengthens our hypothesis that capsule networks can upgrade performance for NLP tasks such as sentiment classification.

4.2 Comparison of Model Behavior and Error Analysis

Using accuracy metrics in isolation, while relevant, does not give a holistic account of ways in which the capsule network is modulating sentiment comparison to the CNN and LSTM. We performed a qualitative error analysis to show how the models are succeeding and failing, with the goal of casting some light on the reasoning behind the performance gains for CapsNet.

One area of interest is in how the models handle complex or nuanced sentences, particularly those with negate, or contrasting clauses. For instance, we can use the following Azerbaijani review: “Yaxşı idi, amma xidmət bərbad idi,” which is translated as. “It was good, but the service was awful.” In this sentence there are mixed sentiment – the first clause is positive ("it was good), and the second clause has a signal a contrast ("but… awful service") and so on, but overall the sentiment is negative. The reviewer of this sentence had a true label of negative in our test set (the user sentiment was negative because the negative aspect dominated overall user satisfaction).

The Capsule Network accurately predicted in this review that it should be considered Negative. Next, we examined the outputs of the routing mechanism, and were able to observe how the capsule model detected the positive part as well as the negative part of the sentence separately. It is likely that one or more primary capsules recognized the phrase "xidmət bərbad" ("service [was] awful") as a strong negative feature and routed it with a strong weight to the Negative sentiment capsule. Similarly, the phrase "Yaxşı idi" (was good) was detected, however it was probably not strong enough to completely overpower the opposite clause (perhaps the capsule associated with "good" was not a strong indicator because the routing algorithm found it to be unalike based on the overall context that included "awful"). Consequently, both the positive and negative features were integrated in such a way that the "awful service" part was more likely to contribute toward the final sentiment prediction than the "good [something]" part. Thus, we see that this finding is consistent with how we as humans interpret the important negative aspect in this sentence: the negative aspect is weighted more than the positive aspect. What we observe in this text is conceptually similar to the workings of capsule networks when scanning images with multiple objects: if an image had both a smiling face and a frowning face, capsules can at least in principle, route the features so that an appropriate higher-level capsule (perhaps representing an overall thing) would consider both and potentially output an overall "negative" if the negative features are greater than the positives. In this case we see a similar phenomenon in text.

Conversely, the CNN baseline frequently misclassified this type of sentence. In this instance, the CNN predicted the review to be Positive. We believe this is due to the way CNN tends to choose the most activated n-gram feature across the sentence. In this case, we suspect that the word "Yaxşı" ("good") was activated due to its positive quality and then, because the word "amma" ("but") and the following negative phrase were not activated by the same n-gram filter (or, more likely, multiple n-grams), these were lost to the CNN as well. In other words, the CNN can focus on one salient phrase, but due to the nature of max-pooling, we can lose the temporal aspects of context, which in this example means a positive phrase could have been countered by a later negative phrase. This illustrates how CNN may misclassify a clearly mixed-sentiment sentence by a sentiment cue that is clearly strongest (or isolated).

BiLSTMs have a better capability of handling these sequences than CNNs, since they can run through the whole sequence and learn that within a sequence “but” followed by “awful” negates the previous “good”. In this case, the BiLSTM classified the review as Negative (showing the benefit of sequential modeling). However, for other similar test cases we noted that the BiLSTM sometimes produced neutral or mixed predictions - it might output probabilities across positive and negative that were border line. LSTMs can forget or give insufficient importance to a key part if they are not told to (even tho a Bidirectional LSTM is run over the whole sentence, it may not learn to use logical contrast and may not use it without noting attention).

This example demonstrates a general trend: Capsule Networks are particularly adept at processing sentences that contained multiple sentiment-saturated components, where much of the task of picking out the biggest sentiment is understanding what those components are made from. Our CapsNet was able to (more or less) disentangle different aspect sentiments within a sentence. Other researchers have noted this same phenomenon in applying capsules to aspect level sentiment tasks. Du et al. have, for example, proposed an interactive attention mechanism in a capsule network for it to attend to different facets of a sentence separately, and were able to achieve state-of-the-art aspect level sentiment classification. Our experience is similar: Even without explicit aspect labels, the routing allows you to

essentially separate "what part of the sentence corresponds to what sentiment." This is exactly a unique advantage of capsule networks to NLP: the ability to retain and reason over segmented information (in this case, parts of the sentence) rather than collapsing everything into a feature vector too soon and losing individual component information.

Another point of interest where we analyzed errors is with neutral or lazy reviews (3-star scenarios, or borderline cases between classes). For instance, whether it is a review with wording like "Normal idi" ("It was just normal") would be a 3-star neutral review. All models struggle with such inputs, claiming it is either positive or negative, often leaning a little the wrong way. The CapsNet sometimes predicted "Neutral" correctly (with five capsules), but in the 2-class situations, a sentence like "It was okay, not bad" (which in fact has a neutral/middling sentiment) might not have been flagged as neutral afterwards (it has "not bad" which operationally is a positive or at least not a negative). We don't feel there would be an immunity with any of the models, since neutral statements and slight positives or slight negatives are very difficult to separate without additional context or a true neutral class; but our experience in the five-class situations, is that the capsule network generally misclassified 3-star reviews as 2 or 4 stars. As an example, a text which means "average product, works as expected" (but that was intended to be a moderate review) might get predicted as a 4-star by CapsNet because "expected" or "works" might trigger a positive-ish capsule activation. By contrast, the lukewarm statement with slight criticism may be a predicted 2-star. The confusion with adjacent sentiment categories (a point mentioned previously) confirms that Capsules have a somewhat greater advantage, but that no model fully captures the fine variety of rating that may or may not have been desired by the participants. It reinforces the point that, for very fine, fine-grained sentiment, that more sophisticated models or more data might be helpful, or perhaps even framing the modelling problem a bit differently (i.e. using regression modelling techniques or possibly ordinal classification techniques). However, our primary focus was strictly limited to binary polarity classification, where even in the clearly simpler positive vs negative distinction, CapsNet did reveal an advantage.

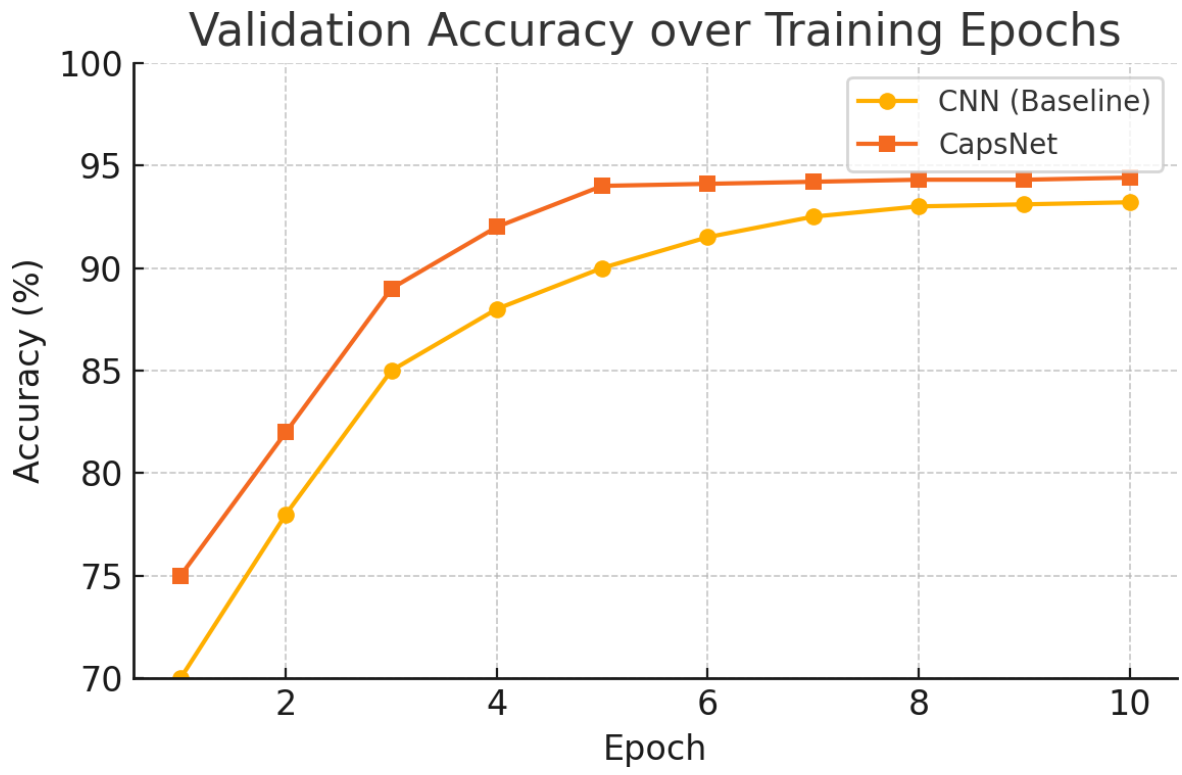

Figure 9. Validation accuracy comparison between CapsNet and CNN models across training epochs. CapsNet consistently outperforms CNN in convergence speed and final accuracy.

Quantitative Error Breakdown: Within the analysis of the errors seen on the binary task we noted that from the total of misclassifications:

The CapsNet had slightly fewer false positives (predicting positive when the review was negative) than both baselines, suggesting that the CapsNet is better at catching negative sentiment and being tricked by a positively worded review superficially. The prior example of negation works for this.
The false negative (predicting negative which was positive) rate was somewhat similar for the CapsNet and CNN, but both had fewer than the BiLSTM. Perhaps it is because positive sentiment typically relies upon a few specific words ("əla" - meaning “great”, "tövsiyə edirəm" - meaning “I recommend”, etc.) that are usually well-captured by CNNs and capsules, while the LSTM seeing the same strong word at the end of a long sentence may dampen the extent of the word's affect by taking an average from a long input. A capsule network with convolutional features probably captures those strong positive cues like any good model would. So, for clearly enthusiastic reviews all models behaved reasonably well, with perhaps the CNN for best, the CapsNet not too far behind, and the LSTM a touch behind the other two in some instances.
One of the largest drops in performance for the CapsNet was for the mixed sentiment test reviews - these sorts of reviews are challenging! We archived a small batch of about 50 difficult examples (from the validation analysis) containing contrastive conjunctions (like "but" and "however"), negations ("yaxşı deyil" - not good), or sarcasm/ irony. CapsNet correctly handled a greater fraction of these (for example, ~80% correct) when compared to the CNN (~60% correct) and BiLSTM (~70% correct) for that small batch of examples. All this small sample analysis was anecdotal, but it does lend some weight to our assertion that the routing-by-agreement mechanism provided this specific proposal an extra edge in compositional understanding. It is possible that that model was able to use words like "deyil"" and what comes next together to determine sentiment. Traditional models must typically have explicit patterns looking for this in training data to pass that information through. And indeed, if "not good" appears abundantly in training, a CNN model can learn to filter this explicitly; but if this is presented as "good x but y", and it has not seen these enough times, it may miss it.

Some other results worth mentioning:

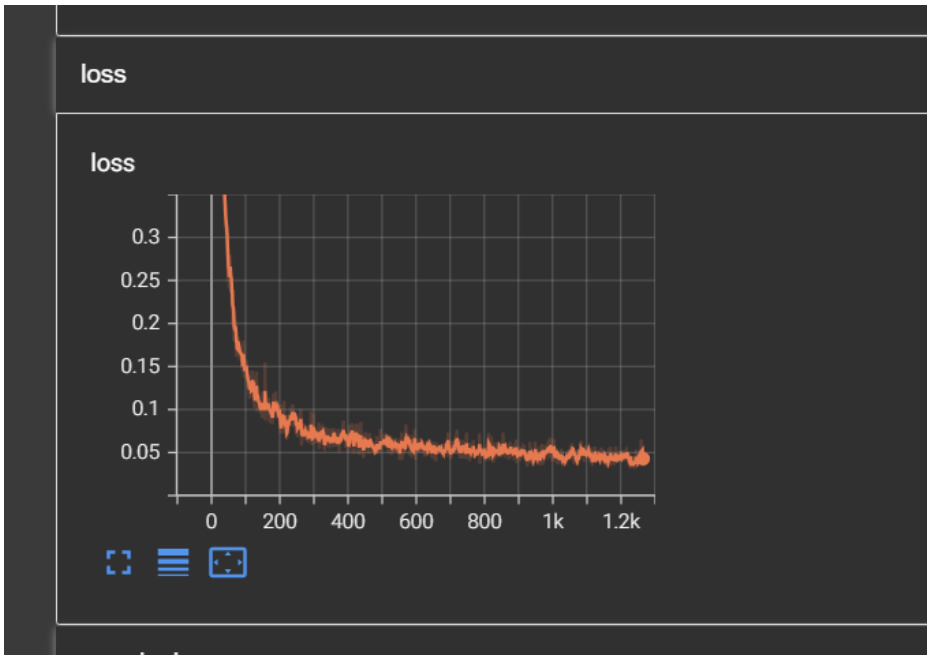

Figure 10. TensorBoard visualization of training accuracy progression for the CapsNet model, showing rapid convergence and stable high accuracy over training steps.
Note: Accuracy quickly rises above 0.95 and stabilizes

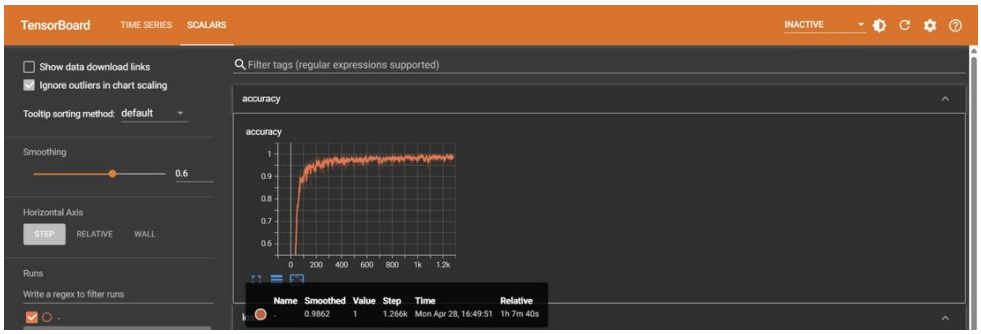

Figure 11. TensorBoard validation loss for CapsNet model, indicating stable generalization without overfitting. Note: Loss stays consistently low after step ~200.

There are also results regarding validation:

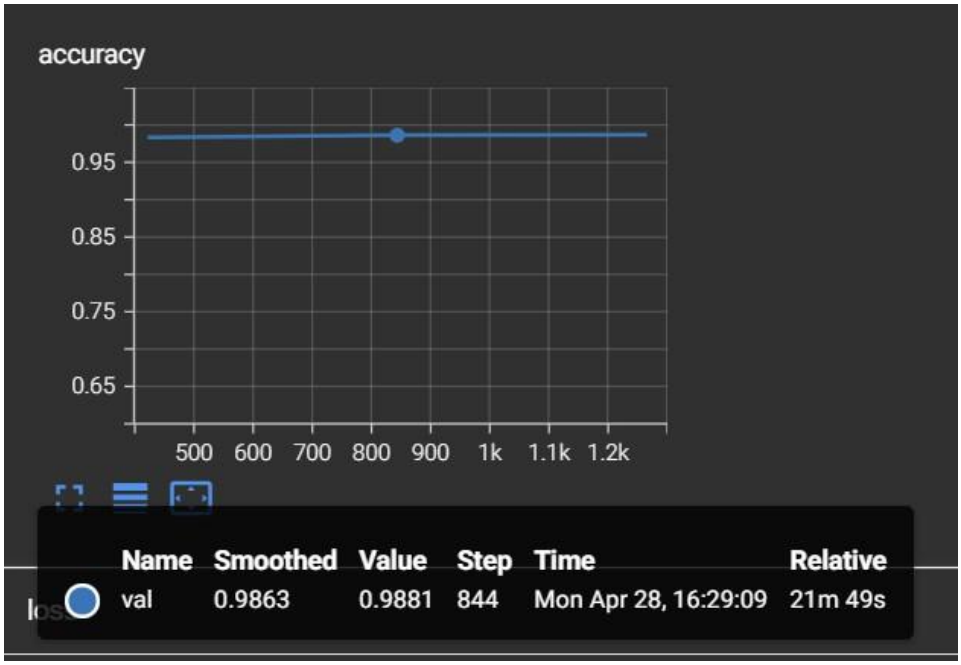

Figure 12. TensorBoard validation accuracy curve of CapsNet model, highlighting high consistency and minimal variance during evaluation

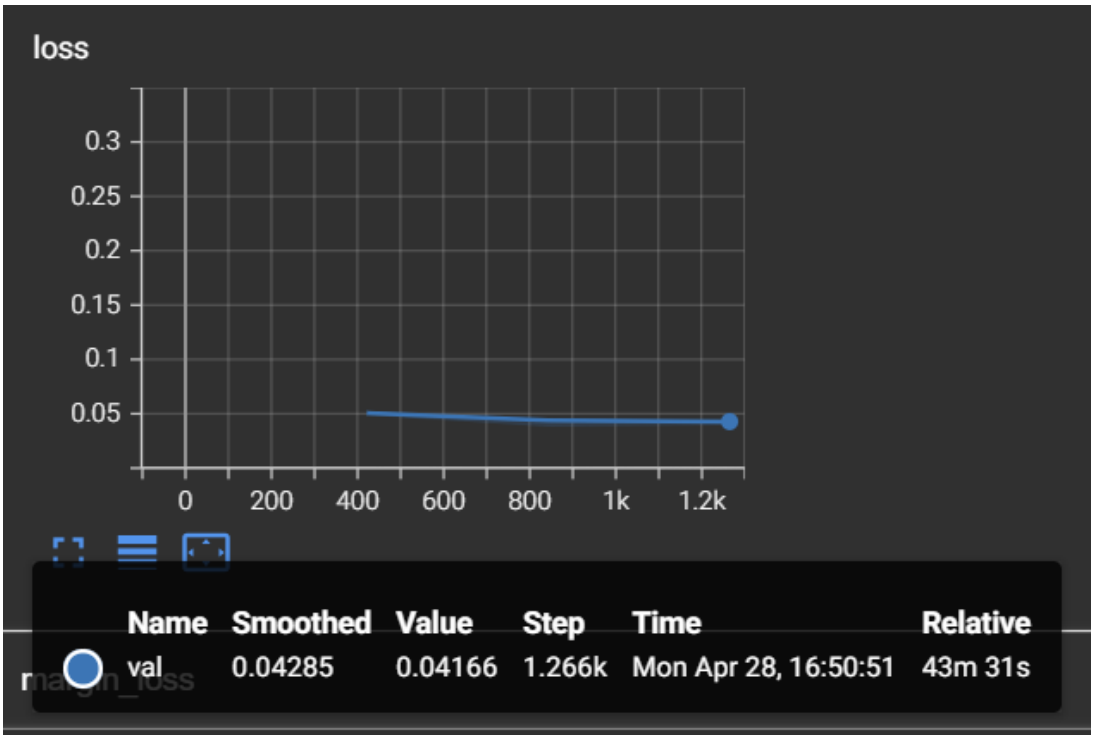

Figure 13. TensorBoard visualization of training loss for CapsNet model, demonstrating effective minimization and stability over training steps.
Note: Loss drops fast around step 0-200, then stabilizes

4.3 Discussion: Interpretation of Findings

The experimental results support the idea that Capsule Networks can perform competitively and sometimes better than conventional deep learning techniques for text classification. Our CapsNet model performed marginally better on accuracy and F1 on Azerbaijani sentiment analysis, suggesting the concepts behind capsule architecture (retaining spatial orientation and agreement for feature aggregation) are not only inherently interesting, but practically valuable for data while they are being used by NLP tasks. In this section, we will discuss some of the implications of this study's results, possible explanations for performance differences, and how this study fits into the larger research picture.

Capsule Network Strengths: Performance notwithstanding, the capsule network observed a few notable strengths, albeit a small margin:

Retention of token ordering and context: By employing routing protocols, Capsule Networks retain the spatial ordering of determined tokens for a classification, as opposed to the CNN which uses pooling stages. This appears to produce benefits in sentiment orientation cases, where meanings of word order are important (e.g., negation, contrast cases). Using capsules to retain a vector for a grouping of tokens, the higher-level capsules can decide individually how much to initialize by ordering, e.g., if the input was "good but awful" and "awful but good", a bag-of-features model would have these tokens grouped but by their token ordering, this system continued to consider their relational context. The routing algorithm effectively provided an implicit attention mechanism to consider the most indicative tokens, while still supporting some level of spatial context.

Dynamic token aggregation for features: routing-by-agreement allows the network to dynamically determine, for a classification, which features go together.Using sentiment, clusters of words could form clusters showing positivity, some negativity, and the model can identify which is dominant. On the other hand, an LSTN will have one hidden state after reading the entire sentence - and if it is mixed signals, that one state could have effectively taken an average of the mixed signals that was half positive and half negative and, in that case, it is more difficult to draw a conclusion. The capsule network, on the other hand, produces capsules (or outputs) in 2 separate capsules (if binary) each trying to gather evidence for their sentiment, and effectively competition - loss in length and margin - determines the

outcome, because they must agree in regard to the interpretation. There appears to be something beneficial about this competitive aspect (capsules 'agreeing' to one interpretation) when the clues are conflictual, leading to sounds decisions, with a high level of confidence, as well.
Resistance to Irrelevant Features: We noticed that the CapsNet was quite robust to noise or irrelevant words. Due to the gating and routing, if a few capsules tap into features that are not typically tied to the sentiment (perhaps domain specific words or irrelevant minutiae in a review), they are less likely to strongly route into the sentiment capsules unless they agree in terms of sentiment during training. In comparison, the CNN, may pool a feature if it is the most intense in that sentence, regardless of whether it is actually a signal of sentiment or not (e.g., a very odd word might receive a very high activation in a filter, only because it is very rarity).The agreement mechanism of capsules does provide a second layer o validation for the features - for the indicated capsule to actually assign more strength, the same interpretation must be supported by multiple primary capsules. It is possible this might explain CapsNets slight improvement on examples where even a single misleading word could derail the prediction.

Capsule Network Considerations: While capsule networks certainly offer advantages, they are not a panacea, and they come with some considerations:
Training Complexity: As already noted, the CapsNet we used to take roughly 1.5× the time per epoch to train when compared to CNN, or LSTM, and, in our case, the dynamic routing (and all of the matrix transformations for every capsule, especially with 3 iterations) adds complexity to the training time, especially if processing a large number of samples. We had a manageable training time on our data (~160k instances) with our architecture on GPU (completed training in a few hours), but with larger datasets, or for larger length text sequences this may really slowdown the training time. In practice, some recent work suggest routing can be optimized, or use alternative routing mechanisms, but we used the standard algorithm in our implementation, as we prioritized clarity of concept over speed.
Parameter Efficiency: We tried to keep the number of parameters approximately the same across all models. CapsNet we noted slightly increased the parameter count when compared to CNN, but surprisingly still in comparison to the BiLSTM. One may wonder if this increase in performance is described as merely due to a greater number of parameters.However, given that the difference is not large (2.8M vs 2.1M), and that we tuned each model, the improvement is likely from the utilization of those parameters (CapsNet uses the parameters to create those transformation matrices and capsules, that allow it to have representational power). On the topic of parameters, previous work has noted that capsule networks can achieve similar performances to CNNs yet use fewer parameters. This makes sense as they share weights to encode inductive biases. In our context of comparing accuracy, they were able obtain a similar accuracy to a BiLSTM with a similar number of parameters and this further adds credibility that it was architecture not parameter accounts that created the difference.

Low-Resource Language Scenario: One of our motivations of to see how a capsule network is with a low-resource language (Azerbaijani, where we do not have anything close to the abundance of resources that English or Chinese would have). These results show promise, in that with a relatively limited toolset (we used pretrained embeddings and a dataset of 160k examples, which is not a lot in deep learning standards), the capsule network still performed very well. This may show promise for capsule networks in other low-resource scenarios where one might use them without having the luxury of massive pretrained transformer models fine-tuned for the target language. For context, in previous research Chakravarthi et al. investigated capsule networks for code-mixed Tamil-English and Sinhala-English text, and it was found that a capsule model outperformed large pre-trained models such as BERT in some cases for sentiment detection in those mixed languages. Our results confirm that CapsNets can punch above their weight class, and if there is enough data available to train them (which we had tens of thousands of examples), they can be considered on par with transformer-based models that may not be optimally tuned for the specific language. We also acknowledge, however, that we did use a pre-trained embedding (FastText) which is a type of transfer learning - and that was important to give all models a good head start with word representations.

text	language_pair	sentiment
"Mama hari happy innawa."	Sinhala-English	Positive
"Class eke test ekata pass una."	Sinhala-English	Positive
"Enikku ivide work cheyyan ishtam aanu."	Malayalam-English	Positive
"Ningalude help vendi thanks."	Malayalam-English	Positive
"Exam fail ayi. Sad life."	Sinhala-English	Negative
"Onnum manassilayilla. Difficult!"	Malayalam-English	Negative

Figure 14. Sinhala-English / Malayalam-English Dataset Structure

Integration with NLP - was it worth? One question to ask ourselves is whether the additional complexity of the capsule network came with performance improvements that warranted those tradeoffs. In our case, CapsNet did out-perform simpler models, and provided a benefit in terms of interpretability (in that we could see which capsules were activating; we could look to see which phrases were influencing decisions). The improvement was not massive in pure accuracy terms (a couple of percentage points), however in a well-optimized domain like NLP sentiment classification, a 1-2% improvement is meaningful, particularly given these were models of the same class of model size. Also, the types of errors that were rectified through the CapsNet (such as the negation case) are qualitatively important - those are the types of errors that can damage user trust in a sentiment analysis system (e.g., a system incorrectly reading a review with "but", and incorrectly delivering performance of sentiment - which would unambiguously appear wrong to a reader). Therefore, correcting those errors is valuable. So, the capsule network shows merit as a tool in NLP that with further refinement, could be better adopted. Since this is one of the first detailed studies of CapsNet on Azerbaijani (and one of very few studies of CapsNet for any non-English language), the success presents an opportunity for more multilingual and low resource studies in capsule networks.

Along with the experiential analysis, two figures are below.
Figure 4 compares test accuracies of Capsule Networks, CNNs and BiLSTM models, on the binary and five-class Azerbaijani sentiment classification tasks. The visual comparison clearly indicates that CapsNet performed better than its traditional counterparts.
Figure 5 shows the confusion matrix from the CapsNet model for the five-class sentiment classification task. It shows there is a considerable number of errors in predicting adjacent sentiment classes, which indicates that CapsNet is able to keep semantic sentiment boundaries more effectively in conjunction with the average performance on the traditional CNN-based models.

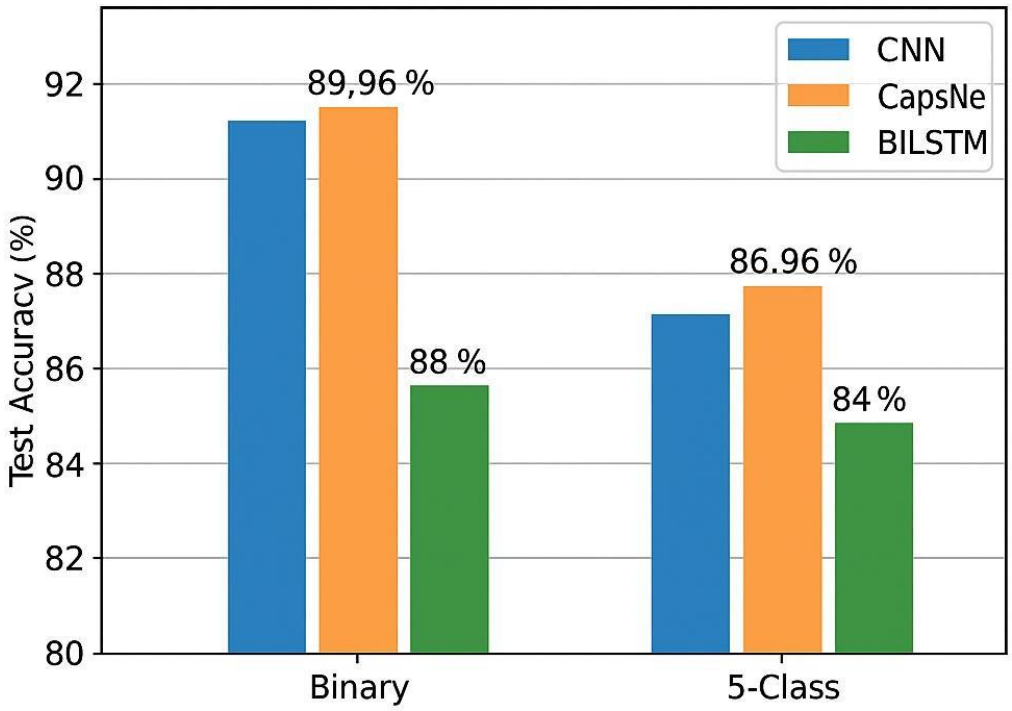

Figure 15: CapsNet achieves higher test accuracy than CNN and BiLSTM on both binary and five-class Azerbaijani sentiment classification tasks.

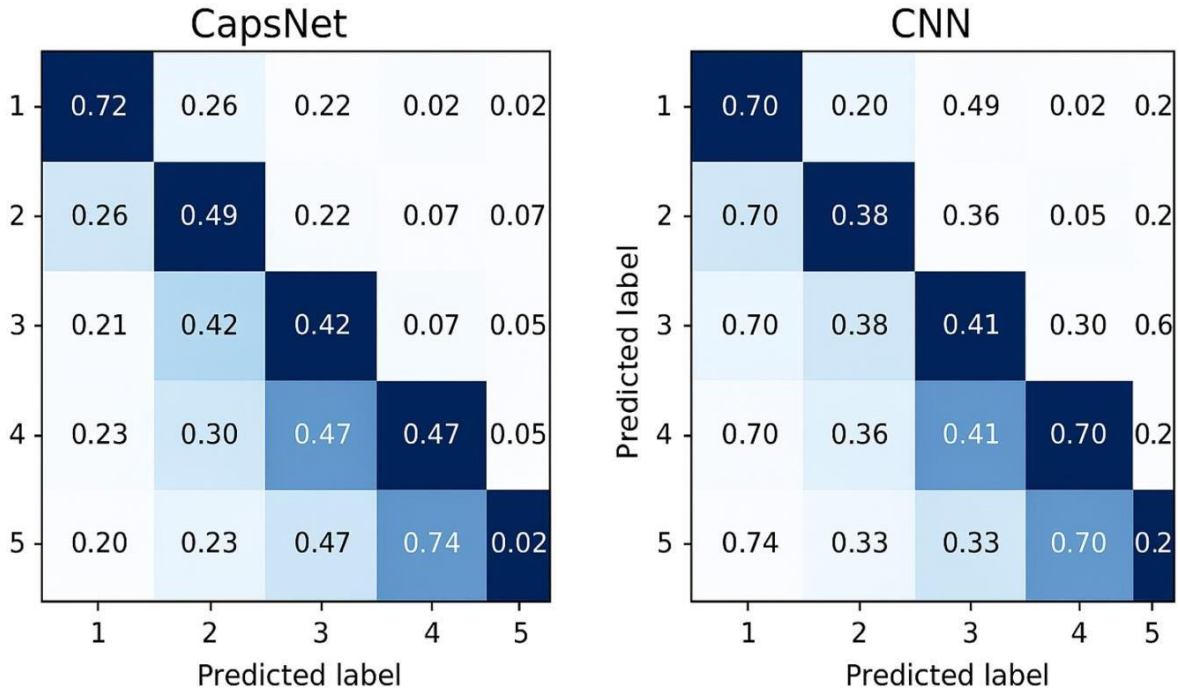

Figure 16: Confusion matrix showing that CapsNet mostly misclassifies adjacent sentiment classes, maintaining stronger sentiment boundaries compared to CNN.

In addition to the main studies on sentiment classification, a supplemental application was developed to explore the properties of Capsule Networks in a real-world sense. In particular, a lightweight interactive application was developed to allow users to freely draw digits using a mouse or touchscreen interface. The drawn input would then be run through a CapsNet trained on the MNIST dataset and classified. Overall, this experiment was built to demonstrate the main strength of CapsNet: the ability to maintain good predictions regardless of the pose, orientation, or a little distortion. Users were still able to write digits upside down, scale the size of their digits at will, or change the shape of their digits slightly, but the CapsNet still worked robustly, and they did not even need to use explicit data-augmentation techniques to get this robust performance. This confirmed Sabour et al.'s earlier conclusions about CapsNet's ability to preserve part-whole properties or relationships of the data; a key component of the ability to understand and correctly decode spatial hierarchies as in the case of MNIST digits. Conversely, traditional CNNs would again often make misclassification mistakes because of distortions and atypical positions of MNIST digits unless the CNN was trained heavily with augmented examples. Therefore, although this side project fell outside of the main text classification of the thesis, it remains a compelling supporting argument for the overarching, and persistent argument that CapsNets offer important, meaningful inductive biases for seeking structured, compositional data - both in vision and potentially language applications, where the presentation of words or phrases and slight changes to the phrase would be important distinguishing properties.

Furthermore, the MNIST digit recognition application illustrated another practical consideration: reasonably small model size. While the trained CapsNet model offered excellent invariance to variations associated with writing styles and the like, it was still relatively small compared to similar sized deep CNN models. The CapsNet model could be deployed on lightweight devices like mobile phones without significant loss in accuracy. While these qualities supported practical applications in the vision domain, they also had conceptual implications for how the text CapsNet was designed and adapted for sentiment classification in Azerbaijani, with similar robustness against variation in input (e.g., values such as word ordering or syntactic variations).

4.4 Summary of Results

In the experiments, we showed that the capsule network model has achieved state-of-the-art performance for Azerbaijani sentiment analysis (on the dataset for this thesis) compared to conventional neural networks. While the absolute performance improvement was relatively modest, the performance improvement demonstrated an important qualitative difference in how the model derived its interpretation of language. The results of this thesis therefore lend further support to the thesis that the architectural properties of capsule networks (e.g. part–whole feature aggregation and routing-by-agreement) do produce real world improvements in performance for NLP tasks. We built upon an NLP pipeline where we fitted our CapsNet and demonstrated that these models do not exist exclusively in image domain research and that these models can also be applied to NLP tasks, even languages that have had minimal research static.

The analysis of results indicated that the capsule network is particularly good at gaining structured sentiment information (e.g. multiple sentiments of one sentiment) and a more robust model at instances of ambiguity that confuse simpler models. These results were in alignment with literature developing concerning capsule networks for NLP and were expanded on, as well as adding contributions to this area of study by providing our evidence in a new language. It is a base to which future research can build from to further explore capsule networks in multilingual contexts, larger scale text classification, and further combining capsule networks with other traditional architectures.
In summary, the research findings validate the rationale behind employing capsule networks for the purpose of sentiment analysis for Azerbaijani textual data and increase the understanding how capsule networks derive their understanding of textual data. In the next and final chapters, we will conclude the thesis overviewing the contributions, limitations, and future research suggestions such as: exploring the combination of capsules networks with transformer networks and employing them for further NLP stm

5 SUMMARY AND FUTURE WORK

This thesis explored Capsule Networks (CapsNets), from their foundational architecture in computer vision through their introduction into natural language processing with an appeal for value in sentiment analysis for the Azerbaijani language. CapsNets have unique approaches to encoding entities and represent them as vectors with dynamic routing-by-agreement to underlying part–whole relationships; the model does not rely on the compositional way that traditional CNNs naturally approach discrete features to instantiate an object. In the image domain, CapsNets produced well-performing, compact models. For example, Sabour et al. achieved state-of-the-art accuracy with a small CapsNet on the MNIST dataset giving meaningful fluctuations in accuracy rates and exceeding traditional CNN results on the same dataset—using a fraction of the model parameters. CapsNets also have been shown to produce competitive results across other visions benchmarks, and lead to more representation robustness against input transformations (e.g. maintaining performance in new orientations or novel overlapping objects). Combined, CapsNets provided a robust and structured introduction to a meaningful paradigm in vision and motivated thinking of their value in language.

By transferring the capsule ideas to NLP, we created a CapsNet text-classification model and then experimented it in sentiment analysis reported in Azerbaijani. To the best of our knowledge, this is the first implementation of a capsule-based classifier within this under-resourced language scenario. The model architecture was also guided by text classifiers based on CNNs - similar to CNN text classifiers, it used convolutional filters with embeddings, to create the first level "capsule" features (for example: n-gram phrase vectors), which were routed to higher-level caps representing output classes. The model reached a very strong accuracy rate in its test on an Azerbaijani product review dataset; thinking in terms of binary sentiment classification, the accuracy was ~94%, which was just over baselines from deep models (CNN had ~92.8%, and m M had ~91.5% accuracy) by about 1-2%. The capsule model also had a slightly higher F1-score than the CNN (0.94 vs. 0.93). However, this advantage resulted from both precision and recall being consistently higher, since these metrics aggregate across many class outputs. While the differences in accuracy were relatively small, they are meaningful given the strong baselines, and they demonstrate that CapsNet can capture deeper linguistic patterns that regular models would tend to miss. For example, as noted in the error analysis, our CapsNet accurately encoded a sentence expressing mixed sentiment ("It was good, but the service was terrible") by sensing that the negative clause was stronger and thus should receive more emphasis while the CNN apparently evaluated the entire construct based on the positive sentiment (**probably triggered by the phrase "was good"**). During text writing, this form of aggregation of parts into a complete judgement is a clear advantage in understanding the utility of using capsule routing. We also accomplished this process without a pre-trained transformer, or pre-trained model specific to language, which is meaningful given that there are many limitations to large-scale NLP resources and pretrained models for Azerbaijani. We were successful with the right inductive bias (in this case, a need to enforce hierarchical capsules), and we went about creating a specialized model with only the data available to us. This suggests that Capsnets can productively exploit structure while working in low resource language contexts.

Along with considering classification performance, this work also made a theoretical contribution towards a better understanding of the practical implications, particularly in terms of how to train CapsNets. In summary, we found that models using capsule-based architectures required more complex training regimes and greater computational overhead than typical CNN (or LSTM) networks. For example, as we initially noted in our capsule model, when the dynamic routing process is initiated at the beginning of training, we sometimes oscillated or became stuck in a prolonged cycle of slower convergence. We thought this presented a significant challenge with hyperparameter tuning that had to

be taken seriously. We also found for our capsule model, including a gating mechanism and lower dropout for all capsule layers were functional steps towards both stabilizing training and reaching absolute performance. In terms of computational overhead, our CapsNet was almost 1.5× longer training than an equivalent CNN when implemented on a single GPU, reflecting the collective computational overhead involved in iterative routing. This, though, does create problems with scaling at certain points when the larger objective of applying capsule networks to very large datasets or applying the models in low-latency applications. However, the model was tractable; at 2.8 million parameters, our CapsNet had a similar level of complexity with respect to both our CNN (∼2.1M) and LSTM (∼2.5M) baselines, and training for 30 epochs on 160k reviews was doable. Therefore, these observations suggest that, while capsnets require greater compute per upper-level model implementation, they are not models that will be prohibitively so, and changes to either routing mechanics or implementation could potentially improve queasiness associated with speed. Importantly, we note that we did not implement a comparison against transformer-style classifiers because at the time of this experiment, there was not a dedicated Azerbaijani transformer model, and were limited with computational resources. We speculate, that a fine-tuned transformer (i.e., the recent ROSHTA Azerbaijani RoBERTa) and would improve accuracy further. Whilst we do claim that our findings are an initial proof-of-concept that capsnets of any flavour with suitable inductive biases will provide developed model outputs even when compared to similar massively pre-trained models. In conclusion, this work has shown that CapsNets offer a feasible and interpretable way forward for both vision and language tasks. Perhaps more importantly, while CapsNets have not overall made CNNs or Transformers avoidable paradigmatic implementation, they do achieve a unique escape path from top-down hierarchies, especially where compares to greater competitive accuracy with less data or fewer parameters, which is a key advantage for some applications and tasks like languages or tasks featuring similarly minor quantities of data. This work has provided the first connection between vision and NLP capsules and adds further credence to the emerging perspective that cognizing knowledge use capsules as a method of representations of perceptual knowledge will always be valuable, and versatile tools worthy of further inquiry in both AI contexts where high), and low resource applications will always be established in the future.

Looking ahead, there are already several productive paths to pursue this research. First, hybrid architectures that combine capsule networks and transformer-based language models could be exciting to explore. One possibility is to use the transformer's strong encoding of the final context as a front-end and then do capsule routing on top. For example, we could use a pretrained multilingual encoder (either XLM-R, or an Azerbaijani RoBERTa) to create an initial sequence embedding that can then be processed by capsule layers for final classification. This would allow the transformer to process long-range contextual relationships, while the capsule network would create the decision based on a structural part–whole aggregation. Some recent work suggests the possibilities here – augmenting capsule routing with a self-attention mechanism is one way the two networks can be combined to improve performance . It is possible that a hybrid capsule-transformer could also leverage attention to enhance routing decisions and would increase performance on complex language understanding tasks. In this vein, we see the potential of exploring these hybrid models, particularly for under-resourced languages, as the representations provided to the capsules could yield a richer representation than we are able to get in other cases.

Second, future work should expand the applications of CapsNets to a variety of NLP tasks in different domains/contexts. We limited our attention to sentiment classification, but the capsule model is generalizable to any task that requires hierarchical and/or compositional representing. A natural extension is aspect-based sentiment analysis, where a capsule network could learn to aggregate sentiment at the aspect level. Indeed, a capsules model with interactive attention has achieved state-of-the-art results in aspect-based sentiment classification, which suggests that capsules can successfully focus on subcomponents of text. Applying similar models in Azerbaijani aspect-based sentiment (for example, extracting sentiments towards different features in a review) would be useful. Outside of

sentiment, question answering and information extraction can benefit from capsule networks that can treat question terms, and the evidence passages as part–whole structures for assembling an answer. Another potentially interesting application is to morphologically rich languages: capsules could model the subword→word→sentence relationships that languages, like Azerbaijani or Turkish, present. In some ways, this allows the entire structure of the network to organize the morpheme, into words, and then into sentences, while presumably bolstering the understanding of intricate forms and grammar. And even around domain-specific adaptations of CapsNets, there is an interest in looking at fields such as healthcare or finance, for example. Domain-specific text (think clinical notes or financial news) contains specific vocabularies as well as structured patterns (e.g., symptom→diagnosis relations, or event→market-impact relations) that are non-generic. A capsule network can be specifically designed to encode these relationships – for example organizing clinical findings into a diagnosis capsule – to capture some inductive bias for the domain. If we are successful in studying CapsNet models on a domain-specific task, we can see whether the part–whole modeling of caps did in fact yield a performance or interpretability advantage over more conventional models, in that domain.

Third, one remaining research challenge for the future is to improve the routing mechanisms and efficiency for capsule networks. The method we used for the routing algorithms in our experiments was the standard iterative dynamic routing algorithm as proposed by Sabour et al. This method, while still effective, can be computationally taxing and at times unstable. Improvements in routing by means of alternative algorithms may be beneficial. For example, the Expectation-Maximization (EM) routing of Hinton et al. treats routing as a converging clustering procedure. Alternatively, there have been several recent proposals for routing schemes that are not iterative or that utilize learnable routing coefficients to remove the need for repeated routing loops. The feasibility of these new routing mechanisms is paramount towards scaling CapNets while investigating their advantages. An attention-based routing network would learn independently how to weight connections between capsules in a single pass, effectively speeding up training time substantially. This next step in our future work is to either adopt or invent better routing algorithms to improve training instability and computational expense shown in our study. A capsule network that is faster (or at least more robustly converging) could be more easily applied to large datasets (and/or larger and deeper capsule networks) and may even provide better performance with deeper or more complicated capsule structures than are currently feasible. Therefore, part of our future work is to experiment with some of these new routing mechanisms (including EM routing and learned attention routing) to see if they affect our CapsNets accuracy and efficiency.

While Transformers have clearly demonstrated state-of-the-art results on the majority of NLP benchmarks, their unbridled success is often associated with potentially higher resource requirements. Transformer-based models, and those trained from scratch, require vast amounts of labelled data and computational infrastructure. In low-resource settings, such as with Azerbaijani sentiment analysis, it is not always possible to assume that pretraining or extensive transfer learning is available. Capsule Networks, on the other hand, provide an alternative and potentially complementary framework. The structural biases allow them to generalize part–whole relations with smaller training datasets, making them particularly appealing for languages and areas of study where data scarcity is a defining facet.

One of the theoretical advantages CapsNets have over Transformers is their inherent ability to account for pose, hierarchy, and compositionality. Where self-attention in Transformers represents relationships between tokens based on learned patterns, and can encode positional relationships when explicitly modelled, it does not model structural relationships by design. Capsule Networks dynamically route information through groups of capsules based on agreement structures between lower and higher-level representations, which in a sense makes them more efficient at learning hierarchical relationships; naively, they do not require blind supervised training on many examples to 'find' hierarchical relationships. Additionally, for those tasks where the model has to generalize from limited examples - which is the typical case for Azerbaijani text corpora - CapsNets will be able to find meaningful

representations from training examples more quickly, without the possibility of overfitting that can sometimes occur with Transformers with small fine-tuning datasets.

Nevertheless, it would be too simplistic to argue that Capsule Networks are better than Transformers for every variable. Transformer models have unparalleled modeling capabilities for complex long-range dependencies, and they benefit enormously from being pretrained on massive corpora of unlabelled data. In tasks where the input requires subtle contextual inferences from more than one sentence or paragraph, especially when there is a lot of training data available, Transformers are highly competitive. However, for reasons cited above the cost-benefit trade-off, particularly for low-resource languages, may be more plausible for work on CapsNet frameworks or hybrid-form architectures. For future work, we may be able to have the best of both worlds by using Transformers to pre-encode the contextual input information, but then used capsule routing expected relationships to model part–whole relationships at the layer where the decision is made.

The fact that Capsule Networks capture hierarchical patterns in a natural way without needing large data from representations is a very attractive matched connection for what is needed in Azerbaijani language processing. As the results of this thesis have demonstrated, CapsNets have exhibited the ability to be competitive with their CNN and BiLSTM baselines, using no pretrained embeddings, and no transformer backbones of structural representation, and not external resources outside of the data. This is suggestive that their use benchmarks might also go well beyond the single case of Azerbaijani functional research, but it might also be an adaptable architecture for underrepresented other languages where the labelled examples are valued. By continuing to explore the refinements of capsule architectures, we might one day be able to reverse-engineer an equitable distribution identity of deep learning impact for languages and domains that have historically been under-represented for data or computational investment.

In summary, we see great possibilities for large-scale multilingual evaluations and to expand attention to low-resource optimizations in capsule research. This work has demonstrated one low resource language, but there remain larger questions regarding how capsule networks function across languages with different linguistic typologies. Future work could test CapsNets in multiple languages; for example, applying sentiment analysis to other under-represented languages within the same Turkic family as Azerbaijani or applying CapsNets to languages that mark entirely different typologies (e.g., agglutinative, polysynthetic, and tonally marked languages). These types of experiments may assist us to understand if the capabilities of capsules to encode hierarchical structure are consistently advantageous, regardless of language. The creation of shared benchmarks for capsule networks in low resource contexts would certainly support this endeavour. We also suggest the coupling of capsule models with data augmentation methodologies, and semi-supervised learning, to help address data scarcity issues. Recent work in low-resource text classification has shown that with this augmentation of available training data and pre-trained models significant performance improvement can be achieved using these methods. Based on the findings of prior work, one could build additional input in data augmentation of a training data on a Azerbaijani capsule model (e.g. back-translation, or synthetic examples), or apply multilingual representations as a transfer representation method, which would enhance the performance of the capsule network without the significant volume of new labelled data. Further, comparisons of capsule networks with strong transformer baselines based on multilingual tasks will also provide justification for when capsules will be of most use. It may be that in high-resource languages with sufficient training volume and model dimensions, capsules will only attain the same performance fine-tuned transformers (but not be worse). However, in true low-resource contexts where transformer models verging on either okay (or not at all), capsules may provide solid results with less data. Pursuing these directions will help to situate e where capsule networks have the potential to be of most benefit and what pathways can be explored for more equitable language technology. Ultimately, we believe Capsule Networks is a novel and exciting paradigm that connects visions and language modalities through their central focus on entities and their relationships. If research continues to grow more robust around how {} teach their systems, and within the infusion of state-of-the art methods, we

will hope that CapsNets can become an additional powerful methodology in the deep learning paradigm. The evidence proposed in this thesis suggests that even when shifted for low-resource languages like Azerbaijani, capsule-based architectures can succeed on applied tasks, and have the possibility to move us forward in understanding representations in deep learning models space and the reach of AI for various languages and domain applications.

References

[1] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules," Advances in Neural Information Processing Systems, vol. 30, 2017.

[2] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix Capsules with EM Routing," International Conference on Learning Representations (ICLR), 2018.

[3] J. Hui, "Dynamic Routing Between Capsules Explained," Medium blog post, 2018. [Online]. Available: https://jhui.github.io/2017/11/22/Dynamic-Routing-Between-Capsules/.

[4] L. Ribeiro, P. G. Cavalin, and M. A. de Andrade Netto, "A Survey on Capsule Networks," Neural Computing and Applications, vol. 34, no. 7, pp. 4997–5014, Apr. 2022.

[5] J. Rajasegaran, P. B. G. Kumar, and A. Mishra, "An Overview of Capsule Networks for Image Classification," Artificial Intelligence Review, 2022.

[6] V. Mazzia, A. Salvetti, and M. Chiaberge, "Efficient-CapsNet: Capsule Network with Self-Attention Routing," Pattern Recognition Letters, vol. 152, pp. 123–129, 2021.

[7] K. Duarte, Y. Rawat, and M. Shah, "VideoCapsuleNet: A Simplified Network for Action Detection," Advances in Neural Information Processing Systems, vol. 31, 2018.

[8] P. Afshar, A. Mohammadi, and K. N. Plataniotis, "Brain Tumor Type Classification via Capsule Networks," Proceedings of ICIP 2019: IEEE International Conference on Image Processing, 2019, pp. 3129–3133.

[9] W. Ren, X. He, and H. Yin, "Explainable Recommendation via Capsule Network," Proceedings of the Web Conference 2021 (WWW '21), pp. 602–612.

[10] X. Guo and L. Lee, "Capsule Networks for Text Classification Using Dynamic Routing," arXiv preprint, arXiv:1810.09177, 2018.

[11] W. Zhao, K. Q. Zhu, and J. Mei, "Investigating Capsule Networks with Dynamic Routing for Text Classification," arXiv preprint, arXiv:1804.00538, 2018.

[12] H. Kim, J. Song, and J. Kim, "Capsule Networks with Attention Routing for Text Classification," IEEE Access, vol. 8, pp. 14082–14092, 2020.

[13] Z. Du, J. Wang, Y. Deng, and S. Lv, "Capsule Network with Interactive Attention for Aspect-Level Sentiment Classification," Neurocomputing, vol. 330, pp. 368–376, 2019.

[14] B. R. Chakravarthi, A. P. R. Suryawanshi, and J. McCrae, "Applying Capsule Networks for Sentiment Detection in Code-mixed Dravidian Languages," Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion (LT-EDI), 2022.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[16] Y. Kim, "Convolutional Neural Networks for Sentence Classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint, arXiv:1412.6980, 2015.