UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ– ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Advanced Informatics and Computing Systems – Software Development and Artificial Intelligence»

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού Και Τεχνητής Νοημοσύνης»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**English Sign Language Recognition through the Utilisation of Convolutional Recurrent Neural Networks** Αναγνώριση της Αγγλικής Νοηματικής Γλώσσας σε Βίντεο με χρήση Συνελικτικών Αναδρομικών Δικτύων
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Todi Martha** Τόδη Μάρθα
Father's name: Πατρώνυμο:	**Todis Dimitrios** Τόδης Δημήτριος
Student's ID No: Αριθμός Μητρώου:	ΜΠΣΠ19051
Supervisor: Επιβλέπων:	**Dionysios Sotiropoulos, Assistant Professor** Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

December 2022/ Δεκέμβριος 2022

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dionysios Sotiropoulos	**Georgios Tsihrintzis**	**Evangelos Sakkopoulos**
Assistant Professor	**Professor**	**Associate Professor**
Διονύσιος Σωτηρόπουλος	Γεωργιος Τσιχριντζής	Ευάγγελος Σακκόπουλος
Επίκουρος Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

UNIVERSITY OF PIRAEUS

School of Information and Communication Technologies

Department of Informatics

Postgraduate Study Program in

Advanced Informatics and Computing Systems - Software Development and Artificial Intelligence

English Sign Language Recognition through the Utilisation of Convolutional Recurrent Neural Networks

by

MARTHA TODI

Supervisor Professor: **Dionysios Sotiropoulos**

Commission: D. Sotiropoulos, G. Tsihrintzis, E. Sakkopoulos

Master's Thesis
Submitted to the Department of Informatics
Of the University of Piraeus
in partial fulfilment of the requirements for the master's degree in
Advanced Informatics and Computing Systems - Software Development and Artificial Intelligence

Piraeus

December 2022

Acknowledgments

For this study, I would like to thank my Supervisor Professor for his patient and his help, which without it could never been finished! Also, a lot of thanks to my parents for any kind of support they gave me, and they always will! Kind thanks to Dimitrios Karagiannis for the technical advice about the device and the psychological support on the very beginning! Also, must not forget my fellow students Angelos, Meggie, Grigoris and Sonia for their help and friendship along our studying years! And of course a lot of thanks to my beautiful blind cat Έρω , who have been given me so much love and courage to keep writing at the long nights.

Abstract

Τα άτομα με προβλήματα ακοής χρησιμοποιούν τη νοηματική γλώσσα (ΝΓ) ως κύρια επικοινωνία. Αυτή η γλώσσα περιέχει κυρίως συνδυασμούς χαρακτηριστικών κινήσεων των χεριών και των εκφράσεων του προσώπου για τον ορισμό των λέξεων και των προτάσεων. Η σχέση μεταξύ της σημασίας μιας πρότασης και των συγκεκριμένων λέξεων είναι -σε ορισμένες περιπτώσεις- περίπλοκη και δυσνόητη, με αποτέλεσμα η εκμάθησή της να μην είναι εύκολη για τους περισσότερους. Τα μαθήματα Νοηματικής γίνονται σε περιορισμένα ινστιτουτα μεγάλων αστικών κέντρων μόνο ,ενώ δεν διδάσκονται σε κανένα δημόσιο σχολείο. Επίσης, πρέπει να αναφερθεί το γεγονός ότι τα άτομα με δυσκολίες όρασης δεν έχουν κανέναν τρόπο να επικοινωνήσουν απευθείας με τους κωφους ανθρώπους. Αυτοι ειναι πολυ βασικοι λόγοι που δημιουργουν την αναγκη μιας αυτοματοποιημένης μετάφρασης της Νοηματικης Γλωσσας σε γραπτές λέξεις και προτάσεις . Η διαδικασία για να επιτευχθεί αυτό μπορεί να γίνει από την τεχνητή νοημοσύνη και τη μηχανική μάθηση και αυτή τη στιγμή η ερευνητική κοινότητα αντιμετωπίζει δυσκολίες, λόγω της έλλειψης μεγάλων συνόλων δεδομένων καθώς και των διαφορών από γλώσσα σε γλώσσα. Ως εκ τούτου, η ανάγκη να συμβάλουμε στην άμεση επικοινωνία της κοινότητας των κωφών είναι μεγάλη και οι ερευνητές προσπαθούν να βρουν μεθόδους και τρόπους για να την καθιερώσουν. Ένα από τα πρώτα ζητήματα είναι η αναγνώριση κίνησης ενός ανθρώπου χρησιμοποιεί ΝΓ από την τεχνητή νοημοσύνη. Ένα άλλο ζητούμενο είναι η αντιστοιχία των κινήσεων των χεριών με τη λέξη. Ο κύριος τρόπος είναι να εκπαιδεύσουμε Τεχνητά Νευρωνικά Δίκτυα και να βελτιώνουμε συνεχώς κάθε πρόβλημα που συναντάμε. Επομένως, η χρήση πολλών διαφορετικών προσεγγίσεων και η δοκιμή των συμπερασμάτων είναι μεγάλης σημασίας. Αυτή η εργασία προσπαθεί να επιτύχει έναν μικρό στόχο σε αυτό το μεγάλο όραμα και να βοηθήσει άλλους ερευνητές να μεταφράσουν τη ΝΓ σε άλλες γλώσσες, με τη βοήθεια ταξινόμησης εικόνων και μοντέλων βαθιάς ακολουθίας

Hearing-impaired people are using Sign Language (SL) as their main communication. This language contains mostly the combined features of hand motions and facial expressions to define the words and the sentences. The relationship between the meaning of a sentence and the specific words is - in some cases - complex and difficult to understand, as a result of which learning it is not easy for most people. Sign language courses are held in limited institutes in large urban centers only, while they are not taught in any public school. Also, it should be mentioned that visually impaired people have no way to communicate directly with deaf people. These are very basic reasons that create the need for an automated translation of Sign Language into written words and sentences. The process to achieve that can be made from AI and machine learning and right now the research community finds difficulties, because of the lack of large datasets and the differences of each language. Therefore, the need to help deaf community in their communications is big and engineers are trying to find methods and ways to establish it. One of the first issues is the motion recognition of a human that communicate in SL by the machine. Another issue is to match the hands motions with the word. The main way is by training

Artificial Neural Networks and improving continuously every matter that come across. So, using a lot of different approaches and testing the conclusions is crucial. This thesis is trying to reach a small goal in this big vision and help other researcher to translate SL in other languages, with the help of image classification and Deep Sequence Models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Neural Networks (ANNs), serving the evolution of Machine Learning (ML) and by extension Artificial Intelligence (AI), are used in more and more fields and their contribute to research for the acquisition of specialized knowledge is increasing. The same applies to diverse applications in information systems that we use every day. One of their usages is in the translation of words between different languages.
Their use does not only concern words but - among other things - also images and audio-visual data. The translation of SL into spoken language certainly cannot be achieved through written language. Observation of human movement, either physically or via video, is required.
This thesis involves the neural system with the process of deep learning algorithms and by correlating minute-long sign language (ESL) videos corresponding to one English language word. The purpose of the project is to train the system resulting in the recognition of the word in videos where a human reproduces it in sign language. For the success of the result, the image classification method was applied to several series of images frames of each video and analyze by Deep Sequence Models, using the architecture model of the CNN and RNN.

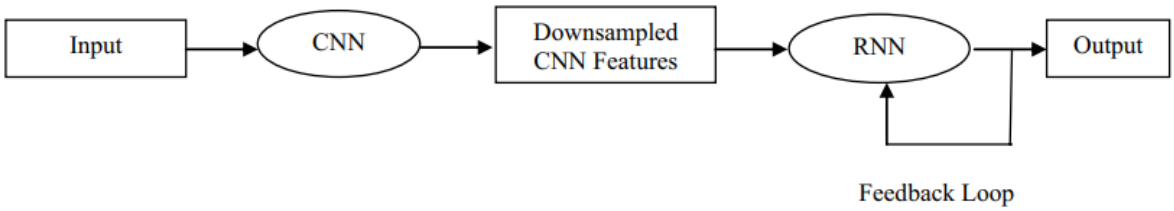


Figure 1 CNN-RNN model

In this project, after an Introduction to Deep Sequence Models- some relevant literature-and a quick overview of the software that has been applied, a detailed analysis of the techniques, the architecture of the model and the code that has been used, will be presented.
More specific, on the next chapters the preparation of the WLASL dataset, as well a guide of data's preprocessing, is written. Furthermore, the architecture model will be discussed, with the code explained in big analysis, almost line by line. To configure out that the architecture and the code are productive, different size of dataset's labels applied and there will be a discussion over the results of those. In the end, there is the conclusion of this study and some future work that would improve the model.

1.1 Sign Language Recognition

The last 20 years, with the evolution of technology a lot of studies and researches have been trying to approach the challenge of the Sign Language recognition by a machine. The first projects were based on visual recognition of the hands. In recent years, it has been very difficult to achieve a "true" machine

image. Therefore, the methods that replaced them were equipped with more hardware to capture the movements of the hands with appropriate sensor devices in marker-less or marker-based setups. The sensors used for these devices have a very difficult task to perform and the accuracy of their readings is limited due to resolution and discrimination capability.

Figure 2 Intelligent Gloves

A very good example is the intelligent glove, as shown on Figure 5, to see the most effective methodology and to find the most effective training set (less information as possible) from sensing element knowledge in electronic system. The data are acquired from glove on the right hand used by a deaf person to increase information of numbers between 1 to 10 in SL. To determine each SL number, a k-NN algorithm is implemented as a classifier.
The proposed method fulfilled the goal of reducing the larger amount of data in the training set and providing high performance when implementing a classifier. In this way, it is possible to store the entire alphabet in sign language. (16)

This approach and some others as well, like a study of hand motion for SL recognition, based on an accelerometer (ACC) and a tan electromyogram (EMG) proposed by Kim, were used to record human movement and, in particular, the movement of the hand (14).
But the above projects are hard to use every day, because not everyone has this kind of technology to help them understand the Sign Language when they need it.
For this reason, researchers are taking a different approach, which is to develop deep-learning algorithms to optimize machine visual recognition for SL.

With the use of Neural Networks and a large dataset of videos , scientists can train a machine to specify a human's hands movement and recognize that a motion is a pattern , which leads to word that can translate it.
Language Recognition of the human movement in a video by neural networks is done by classifying each situation for the analysis and extraction of the information from the movement. The training of the neuron with this process is called image classification (IC) . A video, however, is a continuous situation over time and it is necessary to record and store it, in order to fully understand the action presented. Sequence Models (SM) have been motivated by this scope.
Therefore, the hybrid architecture of deep sequence Models with image classification is quite effective because is needed the spatial information of IC tasks and the temporal from the SM.

1.2 Image classification

An image can convert to groups of pixels or vectors, as well as the multiple frames of a video. Image classification is the task of convolutional neural networks (CNN) that categorize and add labels to them, depending on specific rules and finding the same features. The scope is to recognize what an image represents.
An example of one of the most typical image classifications is face recognition.

For this thesis, the first need is to classify the movement for each world of sign language and localize only the human's hands of the whole picture (spatial processing).

The videos are used for the input layer of the NN and after the process of hidden layers with image classification produce the percentage quantity of the words that are represented. (1)

1.3 Sequence Models

The training information of NNs may be an audio clip, text streams or video clips that represent time sequences of information. As we tend to be mentioned, the training videos for this thesis area unit consist of a sequence of video frames.

When both the input and the output are sequences of data, deep sequential models are utilized. Data points may be grouped into sequences so that observations at one point in the series can be used to infer important information about observations at other points in the sequence. When a sequence is the input and a single data point is the output, such in the cases of video activity identification, emotion categorization, and stock price predictions, managing infinite supervised learning jobs is necessary for handling sequence data. On the other hand, as in speech synthesis, music composition, and picture captioning, the output could be a collection of data points and the input a single data point. Then, the lengths of the input and output sequences may be the same or different in situations where both the input and the output are sequences, such as in speech recognition, natural language comprehension, and DNA sequence analysis. (40)

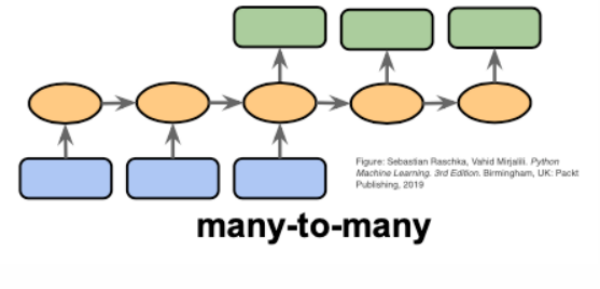


Figure 2 many-to-many

Therefore, the model relates the data across totally different timestamps, in order that will predict the longer-term statistic or to recognize speech/voice or method language.

Sometimes each of the input and output area unit sequences, in some either the input or the output may be a sequence (6)

The types of those models are three. many to 1 (e.g. sentiment classification), One to Many (e.g. image captioning) and many to Many(e.g. Machine translation) and there's another many to many that features Video Classification on frame level a , as shown on Figure 2

Chapter 2

Introduction into Deep Sequence Models

2.1 Convolutional Neural Networks (CNN)

A CNN is an architecture for deep learning that trains directly on data. The scope is to find patterns in frames to recognize specific objects or classify a series of video's frames.
In order to achieve image classification, it is necessary to introduce a CNN layer, which will separate the image into pieces based on filters, which determine the size of the focused area. (2)

Feature Extraction is the part where relevant information from the data is taken and enhanced. It cuts out the redundant information from the region of interest and start taking the features to be used for the classifier calculation. The most common feature extraction is by using the convolution layers of Convolutional Neural Network (CNN) (3)

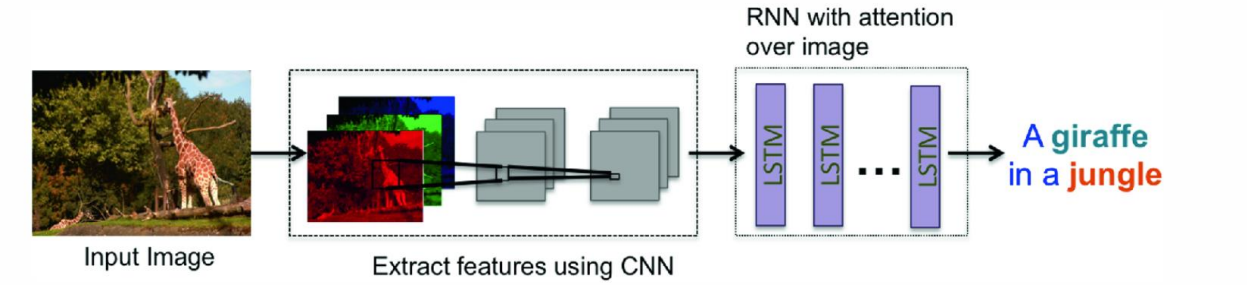


Figure 3 Neural Network Based Image Captioning (5)

Sequence Models have been motivated by the analysis of sequential data such as text sentences, time-series and other discrete sequences data. These models are specially designed to handle sequential information while Convolutional Neural Network are more adapted for process spatial information (4)

If the frame of a video -as input- consists of multiple image inputs as the first layer, which features can be extracted in the next layer. This layer is a whole CNN model that is followed by RNN model – layer to get the sequent data and produced the result of image recognition.
An image of these examples is shown in Figure1.

2.2. Recurrent Neural Network

Recurrent neural network (RNN) may be a style sequence model that has shown economic performance for consecutive information as therein case is video. (7)
The main advantage of victimization RNNs rather than commonplace neural networks is that the options don't seem to be shared. Weights area unit shared across time in RNN. Also, they keep in mind their previous inputs however commonplace Neural Networks don't seem to be capable of memory previous inputs. RNN takes historical info for computation. (8)

RNN can tackle the dealing with sequences of variable length and maintain sequence order. Furthermore, can tracking long-term dependencies as well as sharing parameters in the sequence. These functionalities make the model capable for the training process with videos.

Recurrent Neural Networks are one of the most powerful networks, as a result of the mix of two properties: 1) distributed hidden state that permits them to store plenty of data concerning the past with efficiency, and 2) non-linear dynamics that enable them to update their hidden state in difficult ways in which. (9)

It is possible to unfold RNNs into a standard neural network. In this picture, a single node corresponds to an entire RNN layer. Backpropagation Through Time, which applies backpropagation to this unfolded network, may be used to train the RNN. While RNNs can be taught to capture short-term dependencies between time steps, the so-called "vanishing gradient" problem has made it challenging to train RNNs to capture long-term dependencies. Long short-term memory networks (LSTM) and gated recurrent units, in particular, have been developed as specific sorts of RNNS to address this issue (GRU) (39).The unfolding RNN that ends up being a GRU layer is shown in figure 5.

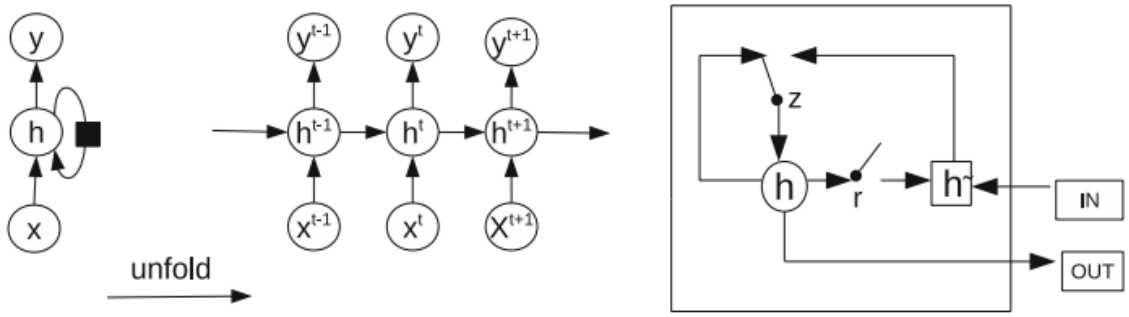


Figure 4 RNN to GRU layers (39)

2.3 Transformer-based model

As Fred Navruzov mentioned "Nowadays, the boundaries between CNN and RNN usage area unit somewhat blurred, as you'll be able to mix those architectures into CRNN for enhanced effectiveness in resolution specific tasks like video tagging or gesture recognition," A transformer could be a deep learning model that provides the flexibility of attention to the training method because it will enhance some elements of the computer file while decreasing different elements -as known as the mechanism of self-attention. This advantage is often exceeded with completely different weights on the dividing computer file. during this case, the hands can the have highest weights, whereas the wall behind the human is 0. Like RNNs, transformers are designed to method sequent computer files, like linguistic communication, with applications towards tasks like translation and text reports. However, in contrast to RNNs, the transformers method the complete input all right away. the attention mechanism provides context for any position within the input sequence and build straightforward to figure parallel, capture the entire image, and eventually to make a decision that half is vital. that's one distinction with RNNs. (11)

The neural transformer network receives an input sentence and transforms it into two sequences: a sequence of word vector embeddings and a sequence of position encodings.
The word vector embeddings are a numerical representation of the text. The words must be converted to the embedding representation in order for a neural network to process them. In the embedding representation, each word in the dictionary is represented as a vector. The position encodings are a vector representation of the position of the word in the original sentence (12).

Chapter 3

Relevant literature on sign language recognition

3.1 Static Sign Language Recognition Using Deep Learning

This project aims to develop a network that can accurately translate a static sign language gesture to its written counterpart, employed Keras and CNN architecture with a variety of layers for data processing and training in order to get specified outcomes*. (*37)
Each of the 16 filters in the convolutional layer has a 2 x 2 kernel. The spatial dimensions are then reduced to 32 x 32 by a 2 x 2 pooling. Convolutional layer filters are raised from 16 to 32, while Max Pooling layer filters are increased from 5 to 5.
Each node in the current layer is randomly disconnected by the Dropout(0.2) algorithm into the next layer.
The model is flattened or changed into a vector, and the dense layer is applied after that. Together with rectified linear activation, the dense layer specifies the completely linked layer.
The SoftMax classifier, which would provide the anticipated probabilities for each class label, was used to complete the model.
The output of the network is a 10-dimension vector for sign language numbers and a 10-dimension vector with 25 dimensions, which correspond to each sign language alphabet. The network was then trained separately for each class using a 35-dimension vector representing static sign language movements.

The system obtained an average of 93.667% based on the gesture recognition in a short amount of time, with testing accuracy of 90.04% in letter recognition, 93.44% in number recognition, and 97.52% in static word identification. A total of 2,400, 50 by 50 photographs of each letter, number, and word motion were used to train each system.

3.2 Hand gesture recognition using a neural network shape fitting technique

In this paper, a new hand gesture recognition technique based on hand gesture features and shape matching neural network is proposed. First, the hand region is isolated using a skin color filtering technique in YCbCr color space. This is a very fast method that results in noise-free segmented images regardless of skin color variation and lighting conditions. The hand shape matching phase and the finger feature extraction phase are based on the innovative and powerful Self-Growing and Self-Organized Neural Gas Network, which approximates the hand morphology in a very satisfactory way . The resulting well-discriminated retrieved finger characteristics are invariant to the size and slope of the hand are factors in a good identification as shown on Figure 2. Finally, the hand gesture recognition, which is based on the Gaussian distribution of the finger features, takes into account the likelihood that all possible finger combinations will match the input hand gesture in addition to the possibility that a finger will belong to each of the five feasible classes. The trials reveal that the identification rate is quite encouraging and is close to 90.45%.

It is important to note that the usage of the SGONG neural network is the primary feature of the suggested hand motion recognition approach. This is due to two factors: first, SGONG is able to represent the form of the hand very effectively, enabling the extraction of robust and useful characteristics; second, it accomplishes this by converging quicker than other networks.

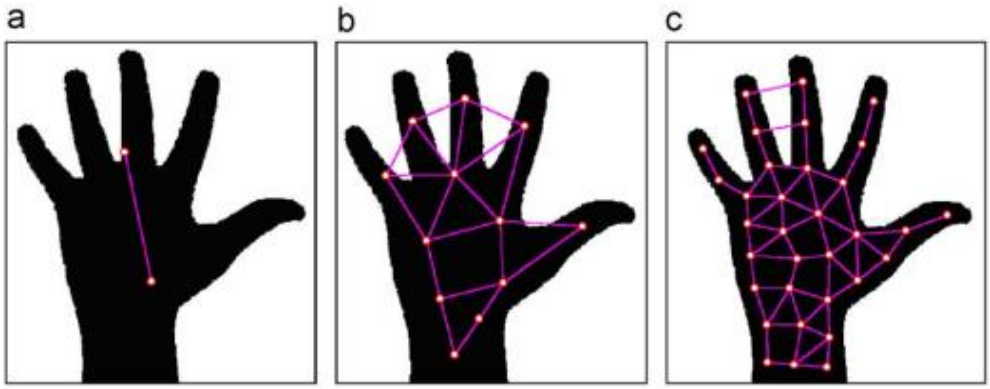


Figure 5 Growth of the SGONG network: (a) starting point, (b) a growing stage and (c) the final output grid of neurons.(36)

With the 15 most relevant features, an average accuracy of 99.82% is achieved for subject-dependent recognition. The generality of the selected features is demonstrated for all subjects with an average accuracy of 96.31%. Unfortunately, this high accuracy does not carry over to subject-independent recognition. (14)

Chapter 4

Technical Environment

Machine Learning and especially Technical Neural Networks needs a strong technical environment. The machine that project is running has NVIDIA GEFORCE GTX 1650 graphic card .

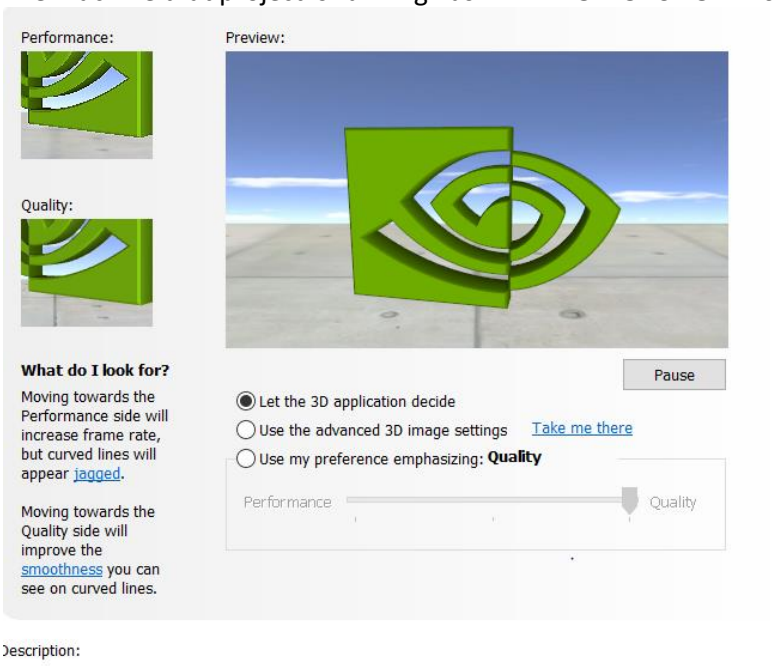


Figure 6 Device's NVIDIA

CUDA

(Compute Unified Device Architecture) is a parallel computing platform from Nvidia Corporation. Provides and API that allows software to use certain types of graphics processing units (GPUs) for general purpose processing, an approach called general-purpose computing on GPUs. CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels. (17)

ANACONDA

The code of the project is written in Python. Anaconda is a good solution to develop machine learning especially in one machine.

Conda

Conda is an open-source package management system and environment management system that runs on Windows, macOS, Linux and z/OS. Conda lets you quickly install, run, and update packages and their dependencies. Conda easily creates, saves, loads, and switches between environments on your local computer. It is designed for Python programs, however it can also package and distribute software for any language.Conda as a package manager helps you discover and install packages. If you would like a package that needs a different version of Python, you do not have to switch to another environment manager, because Conda is additionally an environment manager. With simple a couple of commands, you can set up a completely separate environment to run this other version of Python, whereas still running your usual version of Python in your normal environment (18).

KERAS

Keras is a deep learning API written in Python, that runs on top of the machine learning platform Tensorflow. (19)

It was developed with the aim of enabling rapid experimentation with deep neural networks and it focuses on being user-friendly, modular, and extensible. Keras includes numerous implementations of commonly used neural network building blocks such as layers, targets, activation functions, optimizers, and a variety of tools that make working with image and text data easier and simplify the coding required to write deep neural network code (20)

This project built on Keras because supports CNN and RNN. It supports other common utility layers that is used and explained later, like dropout and pooling .

TENSORFLOW 2.5

TensorFlow is a free end-to-end and open-source software platform that this project using it for training and inference for machine learning and AI .It is an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficient execution of low-level tensor operations on CPU, GPU or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling of computations to many devices, e.g. clusters with hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an easily accessible, highly productive interface for solving machine learning problems, with a focus on modern Deep Learning. It provides essential abstractions and building blocks for developing and delivering machine learning solutions at high iteration speeds (19).

This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Bellow is a warning that appear in the terminal .

```
2022-11-25 00:35:28.954681: I tensorflow/core/platform/cpu_feature_guard.cc:142]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
 to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-11-25 00:35:29.484433: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510]
Created device /job:localhost/replica:0/task:0/device:
GPU:0 with 2143 MB memory:  -> device: 0,
 name: NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00.0, compute capability: 7.5
```

Figure 7 Warning of Tensorflow

These warning informs that the system employs different code based on the hardware as part of going fast. Operations supported by some CPUs that are not supported by others, such as vectorized addition (adding multiple variables at once). Tensorflow is only informing you that the installed version of the software may utilize AVX and AVX2 operations and is configured to do so by default in some circumstances (for example, when doing a forward or back-prop matrix multiplication), which can speed up operations. This is not an error; it is simply informing you that it can and will use your CPU to extract the additional performance.

This project imports NN frameworks that Tensorflow provides, such as Pytoch and Nympy .

Chapter 5

5. Data set Description

The idea of this thesis would never been held if the amazing Database of WLASL isn't exists.
WLASL is the largest video dataset for Word-Level American Sign Language (ASL) recognition, which features 2,000 common different words in ASL . This dataset is Licensed under the Computational Use of Data Agreement (C-UDA) (21).

5.1 WLASL

Machine learning research requires the largest possible number of datasets for training. In particular, in video classification, where videos contain a lot of information, the dataset must contain at least 2000 videos. The WLASL provides 8000 small videos, between 1 and 3 seconds, and each of them corresponds to a word. Thus, each of the 2000 different words is described by 4 videos, each time showing different people with different backgrounds. This gives the model the opportunity to train the neural system with greater accuracy and provide a good sample to recognize the desired part of the video and develop more chances for a successful prediction. (22)

To clone it from git https://github.com/dxli94/WLASL.git
https://github.com/dxli94/WLASL

For download needs the youtube-dll (https://github.com/ytdl-org/youtube-dl)
And running on Linux.

```
@inproceedings{li2020transferring,
 title={Transferring cross-domain knowledge for video sign language recognition},
 author={Li, Dongxu and Yu, Xin and Xu, Chenchen and Petersson, Lars and Li,
Hongdong},
 booktitle={Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern
Recognition},
 pages={6205--6214},
 year={2020}
}
```

Figure 8 license of DB (22)

5.2 Labels preparation

Inside the downloaded file exists a folder where all of the videos has not a name with the world that represents .

Figure 9 The downloaded DB

Also there is a file wlasl_class_list.txt as shown on figure above.

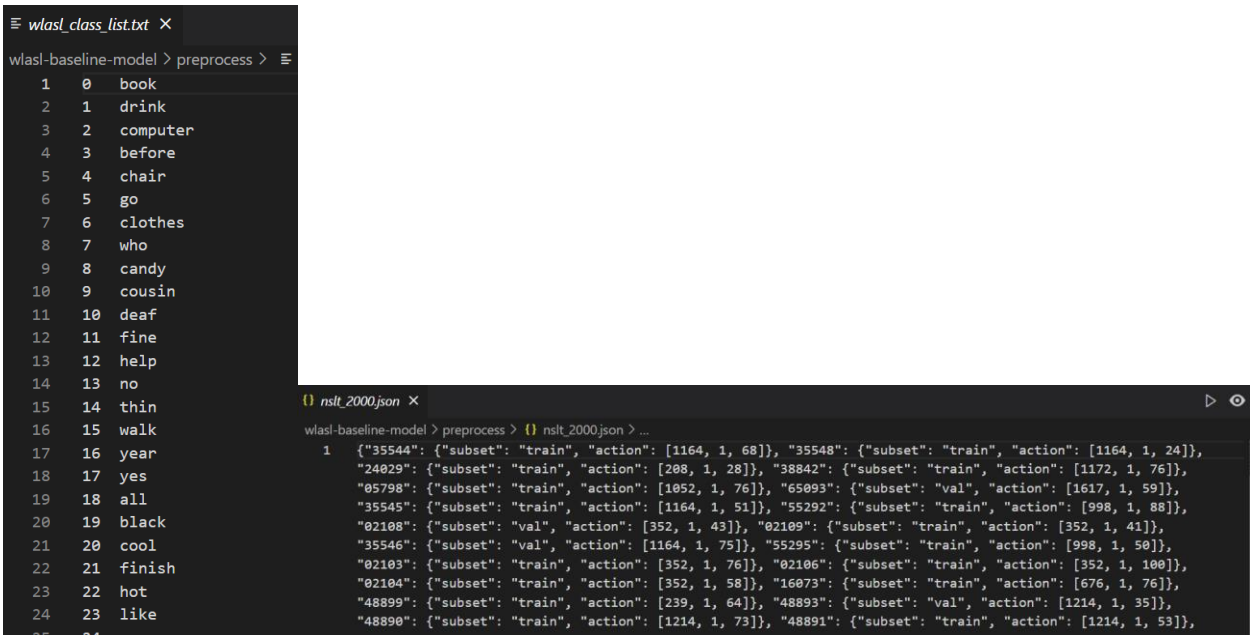


Figure 10 wlasl_class_list and the nslt_2000.json

And a nslt_2000.json , where in action the third value is the number of wlasl_class_list.(Figure 10)

For this reason, has to separate and extract video's names with their corresponding labels. Running the code below,

```
import json
i=0
j=0;
d=''
must=[]
data={}
datafirst={}
datasec={}
with open('preprocess/nslt_2000.json', 'r') as r:
        element = json.load(r)

for v in list(element):
        datafirst.update({v:element[v]['action'][0]})
        with open('preprocess/wlasl_class_list.txt','r') as f:
        line = f.readline()
        for line in f:
                action=line.split()[0]
                translate=line.split()[1]
                datasec.update({action:translate})
```

```
for d in datafirst:
        for s in datasec:
                i=i+1
                j=j+1
                if(str(s)==str(datafirst[d])):
                 data.update({d:datasec[s]})
print(data)
```

Gives the output of the image above which is the desirable json .


```
{'word': 'videos'} preprocess/nslt_2000.json
{'35544': 'me', '35548': 'me', '24029': 'game', '38842': 'nothing', '05798': 'behavior', '65093': 'arizona', '35545': 'me', '55
292': 'structure', '02108': 'alone', '02109': 'alone', '35546': 'me', '55295': 'structure', '02103': 'alone', '02106': 'alone',
'02104': 'alone', '16073': 'diamond', '48899': 'russia', '48893': 'rush', '48890': 'rush', '48891': 'rush', '48894': 'rush', '3
8643': 'normal', '25066': 'good', '25067': 'good', '00854': 'action', '00855': 'action', '25068': 'good', '25069': 'good', '574
83': 'terrible', '53903': 'spirit', '52663': 'smooth', '57484': 'terrible', '27376': 'hide', '27377': 'hide', '27379': 'hide',
'16642': 'disgust', '57486': 'terrible', '11878': 'come', '57489': 'terrible', '08980': 'canoe', '57291': 'tell', '07033': 'bon
e', '19971': 'evidence', '19970': 'evidence', '16709': 'dismiss', '02864': 'apart', '69421': 'one', '16707': 'dismiss', '0633
4': 'bird', '06337': 'bird', '06331': 'bird', '06330': 'bird', '06333': 'bird', '06332': 'bird', '12014': 'common', '12013': 'c
ommon', '02856': 'anyway', '02867': 'apart', '03805': 'assume', '01377': 'afraid', '10709': 'christmas', '48021': 'rich', '1070
3': 'christmas', '10705': 'christmas', '10704': 'christmas', '65507': 'dog', '38648': 'normal', '60915': 'up', '24889': 'goat',
'16646': 'disgust', '26586': 'hard', '26574': 'hard', '45110': 'provide', '45111': 'provide', '45443': 'purple', '05734': 'befo
re', '05730': 'before', '05731': 'before', '05732': 'before', '05733': 'before', '44791': 'professor', '44242': 'pregnant', '44
793': 'professor', '43261': 'plenty', '05229': 'basketball', '26573': 'hard', '42832': 'pink', '42833': 'pink', '42830': 'pin
k', '42831': 'pink', '42836': 'pink', '37790': 'near', '24171': 'gather', '62818': 'weird', '62819': 'weird', '09199': 'care',
'16233': 'dig', '14624': 'dance', '14625': 'dance', '14627': 'dance', '14622': 'dance', '14623': 'dance', '38644': 'normal', '3
8646': 'normal', '05619': 'become', '48897': 'rush', '04388': 'away', '16235': 'dig', '24367': 'german', '37477': 'my', '5417
0': 'square', '54172': 'square', '54173': 'square', '04389': 'away', '04176': 'aunt', '04171': 'aunt', '04170': 'aunt', '1782
```


Figure 11 The output of the json

The final file that must create to be readable from the model has to be csv formatted. So, replacing the ':' with comma and the comma -which separates the worlds – with ';', converts the data like the image above


```
alone;02104.mp4,alone;16073.mp4,diamond;48899.mp4,russia;48893.mp4,rush;48890.mp4,rush;48891.mp4,rush;48894.mp4,
rush;38643.mp4,normal;25066.mp4,good;25067.mp4,good;00854.mp4,action;00855.mp4,action;25068.mp4,good;25069.mp4,
good;57483.mp4,terrible;53903.mp4,spirit;52663.mp4,smooth;57484.mp4,terrible;27376.mp4,hide;27377.mp4,hide;27379.
mp4,hide;16642.mp4,disgust;57486.mp4,terrible;11878.mp4,come;57489.mp4,terrible;08980.mp4,canoe;57291.mp4,tell;
07033.mp4,bone;19971.mp4,evidence;19970.mp4,evidence;16709.mp4,dismiss;02864.mp4,apart;69421.mp4,one;16707.mp4,
dismiss;06334.mp4,bird;06337.mp4,bird;06331.mp4,bird;06330.mp4,bird;06333.mp4,bird;06332.mp4,bird;12014.mp4,
common;12013.mp4,common;02856.mp4,anyway;02867.mp4,apart;03805.mp4,assume;01377.mp4,afraid;10709.mp4,christmas;
```


Figure 13. Reformated Data

Which can be converted to .csv file which contains rows with the name of the videos and their labels(Figure 12)

	A	B	C
1	me,tag		
2	35544.mp4,me		
3	35548.mp4,me		
4	24029.mp4,game		
5	38842.mp4,nothing		
6	05798.mp4,behavior		
7	65093.mp4,arizona		
8	35545.mp4,me		
9	55292.mp4,structure		
10	02108.mp4,alone		
11	02109.mp4,alone		
12	35546.mp4,me		
13	55295.mp4,structure		

Figure 14. Final csv for training

But for memory reasons, the machine can handle max 5000 videos . So, we must take 1000 words-labels that to 5 different videos and separate them to 4000 videos for training and 1000 for testing,with the code bellow.

```
words = [];
#find all the words that have at least 5 videos to be described, from all video dataset
for x in all:
  words.append(x[1]);
fives=[];
for wo in words:
  if words.count(wo) >= 5:
    fives.append(wo)
print("All the words with at least 5 videos")
print(len(fives));
onlyfive = [];
#remove duplicated words from the array above
for  seperate in fives:
   if seperate not in onlyfive:
      onlyfive.append(seperate)

print("Remove duplicated videos")
print(len(onlyfive));
```

```
#Separate the videos-labels to 1000 words and 1000x4 videos for training
#and 1000x1 videos for testing
trainigVideoFour=[];
onlyoneTesting=[];
for s in onlyfive :
    count=0;
    for a in all:
      if (s==a[1]):
       count=count+1;
       if(count<5):
          trainigVideoFour.append(a);
       if(count==5):
           onlyoneTesting.append(a);

print("Video for training")
print(len(trainigVideoFour));
print("Video for testing")
print(len(onlyoneTesting));
```

Chapter 6

Data preprocessing

The layers that exist in this project takes data that must be preprocessed, so that the format can be available to the training model. Especially for a dataset of videos which are ordered sequence of frames and must extract the frames and add them in a 3D tensor. As the size of a video may be differ, so the number of frames changes between each video. The solution is to save the frames of videos at a fixed interval depend on a maximum frame count. If the extracted frames is lesser than this count, the video will be padded with zeros.

6.1. Hyperparameters

First, must define the hyperparameters as shown on Figure 13, where 20 frames must contain each video (Maximum length of the input audio file) and 64 is Batch-size for training and evaluating our model.

```
IMG_SIZE = 224
BATCH_SIZE = 64
EPOCHS = 30

MAX_SEQ_LENGTH = 20
NUM_FEATURES = 2048
```

Figure 15. Defined Hyperparameters

6.2. Build Feature Extractor

The extracted frames have some features that are not useful for training. To extract only the useful parts, an InceptionV3 model from Keras Applications was used. This is a pre-trained network that provides a Keras image classification model

The Inception architecture can lead to high-performance image processing networks that have relatively low computational costs compared to simpler, monolithic architectures

Inception v3 is an image recognition model that has been shown to achieve more than 78.1% accuracy on the ImageNet dataset.

Before the model can be used to recognize images, it must be trained on a large set of labeled images. ImageNet is a commonly used dataset (23).

The model that created is a fully-connected layer at the top, as the last layer of the network with weights of a pretraining on ImageNet, with three (3) Optional shape tuple input ,which width and heigh is 224 and gives a 2D output . As the code of the Figure 14, that is used shows bellow

```
def build_feature_extractor():
    feature_extractor = keras.applications.InceptionV3(
        weights="imagenet",
        include_top=False,
        pooling="avg",
        input_shape=(IMG_SIZE, IMG_SIZE, 3),
    )
    preprocess_input = keras.applications.inception_v3.preprocess_input

    inputs = keras.Input((IMG_SIZE, IMG_SIZE, 3))
    preprocessed = preprocess_input(inputs)

    outputs = feature_extractor(preprocessed)
    return keras.Model(inputs, outputs, name="feature_extractor")

print(np.unique(train_df["tag"].astype(str)));
feature_extractor = build_feature_extractor()
```

Figure 16. builder of feature extractor

Feature maps are generated by applying Filters or Feature detectors to the input frames or the feature map output of the prior layers. Feature map visualization will provide insight into the internal representations for specific input for each of the Convolutional layers in the model. (30)
This project extracts features from the frames of each video, as the code explains above.

What will be returned from the layers that has been grouped into an object, is a Functional API model

6.2.1 Crop Center Square

After extracting rows and columns values from each frame and performing calculations that come from the git code https://github.com/tensorflow/hub/blob/master/examples/colab/action_recognition_with_tf_hub.ipynb , returns the cropped frame.

```
def crop_center_square(frame):
    y, x = frame.shape[0:2]
    min_dim = min(y, x)
    start_x = (x // 2) - (min_dim // 2)
    start_y = (y // 2) - (min_dim // 2)
    return frame[start_y : start_y + min_dim, start_x : start_x + min_dim]
```

Figure 17. Crop Center

6.3 Video Capture

To read frames from videos the OpenCV's VideoCapture method.

VideoCapture is a feature of the openCV library (used for computer vision, machine learning, and image processing) that allows you to work with video by recording it either from a live webcam or from a video file. (24)

The VideoCapture class returns a Video Capture object that we can use to display the video.

```
def load_video(path, max_frames=0, resize=(IMG_SIZE, IMG_SIZE)):
    cap = cv2.VideoCapture(path)
    frames = []
    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            frame = crop_center_square(frame)
            frame = cv2.resize(frame, resize)
            frame = frame[:, :, [2, 1, 0]]
            frames.append(frame)

            if len(frames) == max_frames:
                break
    finally:
        cap.release()
    return np.array(frames)
```

Figure 18. Video Capture

6.4 StringLookUp

The labels of the videos are strings. Neural networks do not understand string values, so they must be converted to numeric form before being fed to the model. Here we use the StringLookup layer, which encodes the class labels as integers. (25)

This layer will cause an error if even one token that used is out of vocabulary and it takes a 1D array that contains all of the translations of videos for train. The output will be the converted input strings to their index in the vocabulary.

```
label_processor = tf.keras.layers.StringLookup(
    num_oov_indices=0, vocabulary=np.unique(train_df["tag"].astype(str))
)
print(len(label_processor.get_vocabulary()))
```

Figure 19. String Look Up

6.5 Prepare All Videos

In the end , calling all the above for the CNN layer.

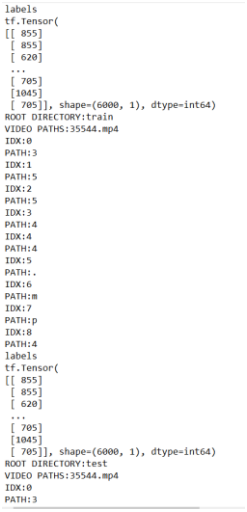
```
def prepare_all_videos(df, root_dir):
    num_samples = len(df)
    video_paths = df["video_name"].values.tolist()[0]
    labels = df["tag"].values
    print('labels')
    print(label_processor(labels[..., None].astype(str)))
    labels = label_processor(labels[..., None].astype(str)).numpy()

    # `frame_masks` and `frame_features` are what we will feed to our sequence model.
    # `frame_masks` will contain a bunch of booleans denoting if a timestep is
    # masked with padding or not.
    frame_masks = np.zeros(shape=(num_samples, MAX_SEQ_LENGTH), dtype="bool")
    frame_features = np.zeros(
        shape=(num_samples, MAX_SEQ_LENGTH, NUM_FEATURES), dtype="float32"
    )

    print("ROOT DIRECTORY:{}".format(root_dir))
    print("VIDEO PATHS:{}".format(video_paths))

    # For each video.
    for idx, path in enumerate(video_paths):
        # Gather all its frames and add a batch dimension.
        print("IDX:{}".format(idx))
        print("PATH:{}".format(path))
        frames = load_video(os.path.join(root_dir, path))
        frames = frames[None, ...]

        # Initialize placeholders to store the masks and features of the current video.
        temp_frame_mask = np.zeros(shape=(1, MAX_SEQ_LENGTH,), dtype="bool")
        temp_frame_features = np.zeros(
            shape=(1, MAX_SEQ_LENGTH, NUM_FEATURES), dtype="float32"
        )

        # Extract features from the frames of the current video.
        for i, batch in enumerate(frames):
            video_length = batch.shape[0]
            length = min(MAX_SEQ_LENGTH, video_length)
            for j in range(length):
                temp_frame_features[i, j, :] = feature_extractor.predict(
                    batch[None, j, :]
                )
            temp_frame_mask[i, :length] = 1  # 1 = not masked, 0 = masked

        frame_features[idx,] = temp_frame_features.squeeze()
        frame_masks[idx,] = temp_frame_mask.squeeze()

    return (frame_features, frame_masks), labels


train_data, train_labels = prepare_all_videos(train_df, "train")
test_data, test_labels = prepare_all_videos(test_df, "test")
print(f"Frame features in train set: {train_data[0].shape}")
print(f"Frame masks in train set: {train_data[1].shape}")
```

Figure 20. Prepare all videos

Now the data is ready for use and the output is on the Figure bellow.

(a)

(b)



(c)

Figure 21.Prepare Videos Results

Chapter 7

Model Architecture

The dataset used for training consists of videos of maximum 3 seconds duration. For successful image classification, each video must be divided into frames that are connected in an ordered sequence. A frame contains 2D images with information about the position of each data relative to other data within those frames. In addition, the ordered sequence frames contain 1D (time) localities. This results in two types of very useful information:
1. Spatial information
2. Temporal information

To model these two aspects, a hybrid architecture is applied, consisting of both convolutions (for spatial processing) and recurrent layers (for temporal processing). (25)
As mentioned in the previous chapter, Convolutional Neural Network spatial processing and Recurrent Neural Network temporal processing can be used.
So it is not difficult to understand why the current hybrid architecture is called CNN-RNN.
Specifically, is been used a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) composed of GRU layers. This type of hybrid architecture is popularly known as CNN-RNN.

7.1 Layers Schematic

In Figure is the a generic Model Architecture , where the first Layers until Batch Normalization, are preprocessing the data- as described on the previous chapter- to create the input to RNN's unfolding layers , which is GRU and the Dropout Layer ,to end as an input for the final layer, which is a Keras Dense layer.

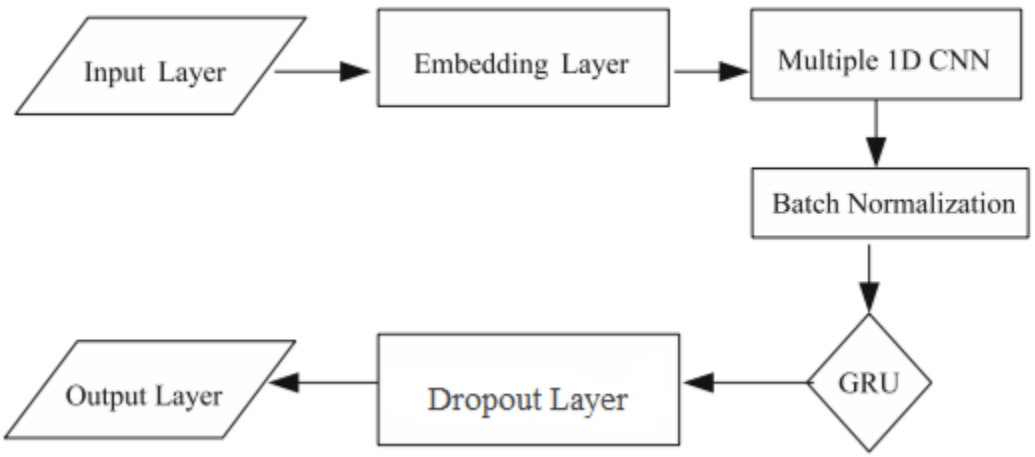


Figure 22 Layers Schematic

7.2 Set Up

```
import numpy
print(numpy.version.version)
import tensorflow as tf
print(tf.__version__)
from tensorflow_docs.vis import embed
from tensorflow import keras
print(keras.__version__)
from imutils import paths

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import imageio
import cv2
import os #from Torch
```

Figure 23. Set Up

PyTorch

The os is for Torch .When we represent data in machine learning, we must do so numerically. A tensor is simply a container that can hold data in multiple dimensions.

Pandas
Is a library for data manipulation and analysis. In particular, it provides data structures and operations for manipulating numerical tables and time series, and builds on Numpy.
Nympy
NumPy is a Python library used for working with arrays. In this project, zeros are used. A new array of the specified shape and type is created and filled with zeros.

7.3 The Sequence Model

7.3.1 GRU

For the problem of the short-term memory first, two GRU layers are created.

Based on accessible runtime hardware and constraints, this layer can select totally different implementations (cuDNN-based or pure-TensorFlow) to maximize the performance. If a GPU is accessible and all the arguments to the layer meet the necessity of the cuDNN kernel (see below for details), the layer can use a quick cuDNN implementation.
(26)
The first layer has an output space dimensionality of 16 and returns the last output in the output sequence. Its input is a symbolic 20-shaped tensor-like object and the corresponding time step is used.

This layer will be the input of another GRU layer, with half dimensionality (27)

```
frame_features_input = keras.Input((MAX_SEQ_LENGTH, NUM_FEATURES))
mask_input = keras.Input((MAX_SEQ_LENGTH,), dtype="bool")

# Refer to the following tutorial to understand the significance of using `mask`:
# https://keras.io/api/layers/recurrent_layers/gru/
x = keras.layers.GRU(16, return_sequences=True)(
    frame_features_input, mask=mask_input
)
x = keras.layers.GRU(8)(x)
```

Figure 24 GRU layer

7.3.2 Dropout Layer

The Dropout layer at random sets input units to zero with a frequency of rate at every step throughout training time, that helps stop overlifting. (28)
This layer takes as input the last GRU layer with a share of 0.4 of its units to drop.

```
x = keras.layers.Dropout(0.4)(x)
```

Figure 25. Dropout Layer

7.3.3 Keras Dense

The dense layer could be a neural network layer that's connected deeply, which suggests every nerve cell within the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the foremost usually used layer within the models.
In the background, the dense layer performs a matrix-vector multiplication. The values utilized in the matrix are literally parameters which will be trained and updated with the assistance of backpropagation.
The output generated by the dense layer is AN 'm' dimensional vector. Thus, dense layer is largely used for ever-changing the size of the vector (29).

In the project, the number of outputs of a layer is 8, using a Relu activation function in the dense layer, which is hidden. This output is the input of another hidden layer, whose size corresponds to the length of the labels of the training videos formatted via StringLookUp and get_vocabulary.

```
x = keras.layers.Dense(8, activation="relu")(x)
output = keras.layers.Dense(len(class_vocab), activation="softmax")(x)
```

Figure 26. Two Dense Layer

7.3.3 RNN model

In the end of building the sequence model taking the output of the last hidden layer.

```
rnn_model = keras.Model([frame_features_input, mask_input], output)
```

Figure 27.Hiden Layer

The training process configuring with compile() , with a crossentropy loss function, which returns a floar tensor and the labels provided as integers . The optimizer of Adam algorithm. Also , the metrics have calculations of the frequency where the predictions equal the labels .

```
rnn_model.compile(
    loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
)
return rnn_model
```

Figure 28. Compile Training Process

7.3.4 Run Experiment

First , the callback for ModelCheckpoint saves ,the latest best in quantity, model's weights in a checkpoint file to be loaded for the training , while showing updates of the starting action.

```
filepath = "./tmp/video_classifier"
checkpoint = keras.callbacks.ModelCheckpoint(
    filepath, save_weights_only=True, save_best_only=True, verbose=1
)

seq_model = get_sequence_model()
```

Figure 29. ModelCheckPoint

The training starts with the fit() which has as input the first 2 columns of trained data and the target is the train_labels, with a 0.3 fraction to the last samples, to evaluate the loss and the metrics in each of the 10 epoch.

```
history = seq_model.fit(
    [train_data[0], train_data[1]],
    train_labels,
    validation_split=0.3,
    epochs=EPOCHS,
    callbacks=[checkpoint],
)

seq_model.load_weights(filepath)
```

Figure 30. Sequence Model's fit

Showing the accuracy of each epoch

```
seq_model.load_weights(filepath)
_, accuracy = seq_model.evaluate([test_data[0], test_data[1]], test_labels)
print(f"Test accuracy: {round(accuracy * 100, 2)}%")
return history, seq_model
```

Figure 31 . Evaluation of Sequence Model

```
Epoch 1/30

2022-11-13 19:40:54.024460: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled
88/88 [==============================] - 41s 314ms/step - loss: 6.9035 - accuracy: 3.5689e-04 - val_loss: 6.9626 - val_accuracy: 0.0000e+00

Epoch 00001: val_loss improved from inf to 6.96259, saving model to ./tmp\video_classifier
Epoch 2/30
88/88 [==============================] - 24s 278ms/step - loss: 6.8762 - accuracy: 3.5689e-04 - val_loss: 7.0270 - val_accuracy: 0.0000e+00

Epoch 00002: val_loss did not improve from 6.96259
Epoch 3/30
88/88 [==============================] - 25s 279ms/step - loss: 6.8528 - accuracy: 3.5689e-04 - val_loss: 7.0909 - val_accuracy: 0.0000e+00
```

Figure 32. Epochs

7.4 Precompute CNN feature map

As mentioned above a CNN generates the rules or data patterns by taking data and results.
This data has to be prepared.
The training process uses the data patterns or rules that have been extracted and gives the trained weights. Based on the learned data patterns, comes the prediction, which provides the class of the frames.

7.4.1 Prepare Single Video

Before starting the training, process must prepare its inputs. Using nympy.zeros which output is a new array of given shape and type, filled with zeros. For the videos is used

```
frames = frames[None, ...]
frame_mask = np.zeros(shape=(1, MAX_SEQ_LENGTH,), dtype="bool")
frame_features = np.zeros(shape=(1, MAX_SEQ_LENGTH, NUM_FEATURES), dtype="float32")
```

Figure 33. Inputs Prepare

Also, each video has to be shorter for the pad, doing the follow.

```
for i, batch in enumerate(frames):
    video_length = batch.shape[0]
    length = min(MAX_SEQ_LENGTH, video_length)
    for j in range(length):
        frame_features[i, j, :] = feature_extractor.predict(batch[None, j, :])
    frame_mask[i, :length] = 1  # 1 = not masked, 0 = masked

return frame_features, frame_mask
```

Figure 34. Videos Shorted

7.5 Model training and inference

In the end, the output of the CNN is the predict action. The labeling process with StringLookup converted to vocabulary form, for a proper print. As the frames and the input of the feature map , on which the training applied. The code is below.

```
def sequence_prediction(path):
    class_vocab = label_processor.get_vocabulary()

    frames = load_video(os.path.join("test", path))
    frame_features, frame_mask = prepare_single_video(frames)
    probabilities = sequence_model.predict([frame_features, frame_mask])[0]

    for i in np.argsort(probabilities)[::-1]:
        print(f"  {class_vocab[i]}: {probabilities[i] * 100:5.2f}%")
    return frames
```

Figure 35. Model Training

After the training a random video has chosen to be tested and saves a GIF of this video to the pc.

```
test_video = np.random.choice(test_df["video_name"].values.tolist())
print(f"Test video path: {test_video}")
test_frames = sequence_prediction(test_video)
to_gif(test_frames[:MAX_SEQ_LENGTH])
```

Figure 36. Last video

Chapter 8

Experimental Results

The training set have enough video for a good training and of course testing.
For memory reasons, as mentioned above, the max length of training and testing video must be 5000.
Running the code with different number of words that have been tested

8.1 101 words

Total videos for training: 404
Total videos for testing: 101
Test accuracy: 0.99%
Test video path: 65059.mp4

The test video is marked with yellow color to the tables below.

almost:1.33%	announce:1.07%	australia: 1.00%	bad: 0.93%	already: 0.87%
apartment:1.28%	animal: 1.07%	bored: 0.99%	approach:0.93%	better: 0.87%
behind: 1.26%	arrogant: 1.06%	attract: 0.99%	baptize: 0.93%	attitude: 0.87%
bake: 1.25%	approve: 1.05%	bald: 0.99%	black: 0.93%	appropriate:0.86%
baseball: 1.25%	also: 1.05%	attention: 0.99%	bath: 0.93%	apple: 0.86%
ask: 1.24%	alone: 1.05%	appointment:0.99%	basketball:0.92%	below: 0.86%
beautiful: 1.24%	beginning:1.05%	baby: 0.99%	away: 0.92%	bee: 0.85%
bet: 1.23%	bar: 1.05%	blanket: 0.98%	bite: 0.92%	beard: 0.84%
balance: 1.23%	aunt: 1.04%	avoid: 0.98%	become: 0.91%	anniversary:0.84%
area: 1.18%	angle: 1.04%	bless: 0.97%	banana: 0.91%	belief: 0.83%
bicycle: 1.16%	believe: 1.04%	awful: 0.97%	ball: 0.91%	back: 0.82%
benefit: 1.15%	behavior: 1.04%	angel: 0.97%	america: 0.90%	bathroom: 0.81%
angry: 1.14%	answer: 1.03%	apostrophe:0.97%	bear: 0.90%	artist: 0.81%
biology: 1.12%	arm: 1.03%	blame: 0.97%	august: 0.90%	arrive: 0.80%
best: 1.11%	analyze: 1.02%	bitter: 0.96%	bird: 0.90%	argue: 0.80%
blue: 1.10%	barely: 1.02%	belt: 0.95%	big: 0.90%	appear: 0.80%
before: 1.09%	balloon: 1.02%	birth: 0.95%	backpack: 0.90%	
bacon: 1.07%	audience:1.02%	blind: 0.95%	because: 0.88%	
bed: 1.07%	borrow: 1.02%	bark: 0.95%	asia: 0.88%	
bell: 1.07%	art: 1.01%	body: 0.94%	awkward: 0.88%	
	assist: 1.00%	bedroom: 0.94%	boat: 0.88%	
		beer: 0.94%	autumn: 0.87%	

8.2. 250 words

Total videos for training: 1000
Total videos for testing: 250
Test accuracy: 0.4%
Test video path: 07266.mp4

2022-11-13 20:42:16.297698: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8201

2022-11-13 20:42:20.252695: W tensorflow/core/common_runtime/bfc_allocator.cc:272] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.99GiB with freed_by_count=0. The caller indicates that this is not a failure but may mean that there could be performance gains if more memory were available.

bored: 0.57%	close: 0.49%	august: 0.45%

couch: 0.56% big: 0.56% bite: 0.55% bacon: 0.55% carrot: 0.54% area: 0.53% correct: 0.53% constitution: 0.53% bless: 0.53% banana: 0.52% camp: 0.52% coat: 0.52% curious: 0.52% australia: 0.52% awful: 0.52% behind: 0.51% calm: 0.51% animal: 0.50% best: 0.50% brave: 0.50% assist: 0.49% cheat: 0.49% body: 0.49% celebrate: 0.49%	buy: 0.49% bed: 0.48% arm: 0.48% bowl: 0.48% color: 0.48% already: 0.48% believe: 0.48% asia: 0.47% bald: 0.47% bread: 0.47% compare: 0.47% beer: 0.47% apartment: 0.46% bottle: 0.46% control: 0.46% calculate: 0.46% cut: 0.46% boyfriend: 0.46% chair: 0.46% bottom: 0.46% catholic: 0.46% barely: 0.46% blame: 0.45% daily: 0.45% appropriate: 0.45%	bear: 0.45% cancel: 0.45% classroom: 0.45% beautiful: 0.45% break: 0.44% cute: 0.44% bathroom: 0.44% because: 0.44% balloon: 0.44% apostrophe: 0.44% conflict: 0.44% common: 0.44% corn: 0.44% comb: 0.44% cop: 0.43% candle: 0.43% autumn: 0.43% below: 0.43% beard: 0.43% boy: 0.43% cow: 0.43% bother: 0.43%

and the other word's estimation until
chocolate: 0.27%

8.3. 500 words

Total videos for training: 2000
Total videos for testing: 500
Test accuracy: 0.2%
Test video path: 11471.mp4

2022-11-13 22:01:50.985051: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8201

2022-11-13 22:02:00.858314: W tensorflow/core/common_runtime/bfc_allocator.cc:272] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.71GiB with freed_by_count=0. The caller indicates that this is not a failure, but may mean that there could be performance g bains if more memory were available.
2022-11-13 22:02:00.886959: W tensorflow/core/common_runtime/bfc_allocator.cc:272] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.99GiB with freed_by_count=0. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.

behavior: 0.23%	cross: 0.22%	because: 0.21%	change: 0.21%	decorate: 0.21%
ball: 0.23%	artist: 0.22%	disappear: 0.21%	deduct: 0.21%	cook: 0.21%
cabinet: 0.23%	double: 0.22%	eagle: 0.21%	banana: 0.21%	bully: 0.21%
announce: 0.23%	biology: 0.22%	delicious: 0.21%	hospital: 0.21%	confront: 0.21%
bottom: 0.23%	head: 0.22%	curse: 0.21%	blame: 0.21%	cafeteria: 0.21%
category: 0.22%	already: 0.22%	deep: 0.21%	carrot: 0.21%	evidence: 0.21%
appropriate: 0.22%	butter: 0.22%	down: 0.21%	dirt: 0.21%	grapes: 0.21%
educate: 0.22%	california: 0.22%	apartment: 0.21%	classroom: 0.21%	chemistry: 0.21%
egypt: 0.22%	cheap: 0.22%	destroy: 0.21%	belt: 0.21%	duck: 0.21%
become: 0.22%	guide: 0.22%	caption: 0.21%	dance: 0.21%	balloon: 0.21%
correct: 0.22%	clock: 0.22%	arrive: 0.21%	describe: 0.21%	expensive: 0.21%
basketball: 0.22%	carry: 0.22%	diploma: 0.21%	beard: 0.21%	fight: 0.21%
asia: 0.22%	brave: 0.22%	bread: 0.21%	early: 0.21%	area: 0.21%
exaggerate: 0.22%	building: 0.22%	build: 0.21%	heavy: 0.21%	if: 0.21%
chat: 0.22%	belief: 0.22%	angel: 0.21%	depend: 0.21%	child: 0.21%
eight: 0.22%	can: 0.22%	brother: 0.21%	bad: 0.21%	girl: 0.21%
decide: 0.22%	bath: 0.22%	convince: 0.21%	guitar: 0.21%	diarrhea: 0.21%
beautiful: 0.22%	discuss: 0.22%	celebrate: 0.21%	develop: 0.21%	catholic: 0.21%
bald: 0.22%	contact: 0.22%	desk: 0.21%	green: 0.21%	enjoy: 0.21%
emotion: 0.22%	infection: 0.22%	ear: 0.21%	bless: 0.21%	baptize: 0.21%
demonstrate: 0.22%	diaper: 0.22%	help: 0.21%	choose: 0.21%	baseball: 0.21%
bitter: 0.22%	fix: 0.22%	cousin: 0.21%	almost: 0.21%	christian:
big: 0.22%	deaf: 0.21%	bite: 0.21%	complain: 0.21%	
energy: 0.22%	electrician: 0.21%	common: 0.21%	cost: 0.21%	
avoid: 0.22%	finish: 0.21%	degree: 0.21%	bowl: 0.21%	
both: 0.22%	bowling: 0.21%	check: 0.21%	calm: 0.21%	
dessert: 0.22%	challenge: 0.21%	function: 0.21%	dig: 0.21%	
ceiling: 0.22%	environment: 0.21%	french: 0.21%	black: 0.21%	
command: 0.22%	coconut: 0.21%	blind: 0.21%	chase: 0.21%	
	bracelet: 0.21%	contribute: 0.21%	art: 0.21%	
	dry: 0.21%		bird: 0.21%	
			house: 0.21%	

	give: 0.21% brown: 0.21% below: 0.21%	bottle: 0.21% event: 0.21% alone: 0.21% cemetery: 0.21% corn: 0.21%	engineer: 0.21% dollar: 0.21% get: 0.21% complex: 0.21% divide: 0.21% family: 0.21% careful: 0.21% congress: 0.21% barely: 0.21% answer: 0.21% dad: 0.21% class: 0.21% east: 0.21%	0.21% home: 0.21% balance: 0.21% doubt: 0.21% attention: 0.21% best: 0.21% escape: 0.21% curriculum: 0.21% city: 0.21% dream: 0.21% coat: 0.20% grow: 0.20%

and the other word's estimation until
fancy: 0.17%

8.4 1001 words

Total videos for training: 4004
Total videos for testing: 1001
Test accuracy: 0.1%
Test video path: 23667.mp4

2022-11-13 19:53:33.100656: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8201

2022-11-13 19:53:42.897532: W tensorflow/core/common_runtime/bfc_allocator.cc:272] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.43GiB with freed_by_count=0. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.

pillow: 0.12% anniversary: 0.12% never: 0.12% egypt: 0.12%	duck: 0.11% analyze: 0.11% door: 0.11% honest: 0.11% piano: 0.11%	bake: 0.11% money: 0.11% coffee: 0.11% choose: 0.11% opposite: 0.11%	community: 0.11% gossip: 0.11% dirty: 0.11% crab: 0.11%	establish: 0.11% christian: 0.11% image: 0.11% picture: 0.11% bother: 0.11%

opinion: 0.12%	old: 0.11%	fat: 0.11%	tired: 0.11%	because: 0.11%
play: 0.12%	heaven: 0.11%	assist: 0.11%	mountain: 0.11%	rain: 0.11%
kill: 0.12%	attitude: 0.11%	glasses: 0.11%	over: 0.11%	drawer: 0.11%
cop: 0.12%	diarrhea: 0.11%	dinosaur: 0.11%	drop: 0.11%	limit: 0.11%
dissolve: 0.12%	insurance: 0.11%	hearing: 0.11%	eye: 0.11%	blue: 0.11%
pass: 0.12%	clock: 0.11%	most: 0.11%	forest: 0.11%	environment: 0.11%
less: 0.12%	box: 0.11%	during: 0.11%	photographer: 0.11%	iran: 0.11%
curse: 0.11%	cafeteria: 0.11%	enjoy: 0.11%	dirt: 0.11%	answer: 0.11%
check: 0.11%	knife: 0.11%	hamburger: 0.11%	jail: 0.11%	grapes: 0.11%
forever: 0.11%	delay: 0.11%	draw: 0.11%	rob: 0.11%	delicious: 0.11%
game: 0.11%	march: 0.11%	cemetery: 0.11%	listen: 0.11%	cry: 0.11%
ball: 0.11%	meet: 0.11%	candy: 0.11%	character: 0.11%	bless: 0.11%
measure: 0.11%	also: 0.11%	father: 0.11%	helicopter: 0.11%	car: 0.11%
fail: 0.11%	friendly: 0.11%	laugh: 0.11%	electrician: 0.11%	exaggerate: 0.11%
no: 0.11%	fish: 0.11%	dive: 0.11%	overwhelm: 0.11%	belt: 0.11%
niece: 0.11%	coconut: 0.11%	many: 0.11%	meat: 0.11%	buy: 0.11%
dream: 0.11%	challenge: 0.11%	letter: 0.11%	flute: 0.11%	plant: 0.11%
goal: 0.11%	boat: 0.11%	describe: 0.11%	jacket: 0.11%	jealous: 0.11%
necklace: 0.11%	guide: 0.11%	name: 0.11%	divorce: 0.11%	exchange: 0.11%
commute: 0.11%	near: 0.11%	diploma: 0.11%	europe: 0.11%	before: 0.11%
keep: 0.11%	conflict: 0.11%	god: 0.11%	skin: 0.11%	list: 0.11%
easy: 0.11%	mind: 0.11%	one: 0.11%	cheap: 0.11%	chicken: 0.11%
helmet: 0.11%	cracker: 0.11%	lawyer: 0.11%	sunday: 0.11%	choice: 0.11%
backpack: 0.11%	example: 0.11%	class: 0.11%	france: 0.11%	tomorrow: 0.11%
inform: 0.11%	memorize: 0.11%	bright: 0.11%	paint: 0.11%	multiply: 0.11%
boss: 0.11%	classroom: 0.11%	love: 0.11%	fly: 0.11%	like: 0.11%
brown: 0.11%	close: 0.11%	head: 0.11%	curious: 0.11%	australia: 0.11%
negative: 0.11%	improve: 0.11%	kiss: 0.11%	how: 0.11%	office: 0.11%
bedroom: 0.11%	hair: 0.11%	area: 0.11%	clean: 0.11%	chemistry: 0.11%
early: 0.11%	confront: 0.11%	jesus: 0.11%	manager: 0.11%	blanket: 0.11%
motorcycle: 0.11%	invest: 0.11%	kitchen: 0.11%	drive: 0.11%	approach: 0.11%
glass: 0.11%	discover: 0.11%	myself: 0.11%	carrot: 0.11%	christmas: 0.11%
angry: 0.11%	friend: 0.11%	need: 0.11%	make: 0.11%	bed: 0.11%
law: 0.11%	past: 0.11%	pay: 0.11%	almost: 0.11%	energy: 0.10%
bully: 0.11%	city: 0.11%	plus: 0.11%	lock: 0.11%	neck: 0.10%
enter: 0.11%	canada: 0.11%	alone: 0.11%	bath: 0.11%	disagree: 0.10%
every: 0.11%	earring: 0.11%	bee: 0.11%		bacon: 0.10%
dog: 0.11%	lecture: 0.11%	discuss: 0.11%		
feed: 0.11%				

			candle: 0.11% college: 0.11% awful: 0.11%	front: 0.10%

and the other word's estimation until
thailand: 0.08%

Chapter 9

Discussion

The results of the previous chapter show that, with a small amount of the words, the accuracy is much better and the 65059.mp4 represents the word 'also' which has 1.05% possibility on prediction.
One of the most important reasons is the number of videos for each word. This code has been tested on another dataset, where there are only 5 labels, but the training videos are 594 and the testing 224. This ratio of labels and their videos gives an accuracy 17.4% and the biggest percentage of prediction to the right tested video's tag. Reminding, that this project's dataset's ratio of labels-videos is 1:4.
In addition , should be noted that the model on the output of the results appearing a warning about the lack of the device's memory , before the final training results of its predictions. This would be another good reason that explains the low accuracy due to the big quantity of videos and the available sources of the device.

```
2022-11-13 20:21:54.707681: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded
cuDNN version 8201
2022-11-13 20:22:00.737576: W tensorflow/core/common_runtime/bfc_allocator.cc:272]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.99GiB with freed_by_count=0.
The caller indicates that this is not a failure, but may mean that there could be performance gains
if more memory were available.
```

Figure 37. Memory Warnings

9.1 Different Ratio of Labels: Videos

Additionally, there were more testing on different ratio, to verify that the above conclusion is accurate. Must be noted that as the number of videos for one word increasing the test and train videos are getting fewer.
Above is the table of these results.

Ratio Word: Videos	Words	Train/Test Accuracy	Training Videos	First Predicted Video	Position of the testing video
1:8	78	1.28%	546	country: 1.93%	44th

					police: 1.17%
1:9	44	2.27%	352	delay: 2.91%	6th yesterday:2.64%
1:10	22	4.55%	198	mother: 5.62%	4th son: 5.07%

On the table can see that as the ratio increasing, the position of testing video is getting higher close to the first predicted video and the accuracy is also increasing.
This approves the hypothesis, that the results depend on the lack of the same videos per word-label in the dataset.

Chapter 10

Conclusions and future work

10.1 Conclusions

In conclusion, the problem of Sign Language recognition from a video by NN , through CNN-RNN architecture, can separate in two basic goals.
The first is the recognition of the Human motion and this study approached it by cutting the video into frames, applying Filters and Feature detectors to the input frames and converting the labels of them into numeric values, with information about the position of each data relative to other data within those frames.
This preprocess is successfully done in this study, and it is important to start the second and main phase. This phase is the training of the model to classify the motions of the video to the right labels. This is a sequence-to-sequence problem and has been solved with the GRU and Dropout layer, where the temporal information has been extracted and used it for the prediction process.
The training process has also been done successfully but the results are not so satisfactorily.

After the discussion, the cases arise that with a bigger dataset and a better ratio of videos for one word, this project would have at least 20% accuracy. In the next chapter will be discussed further improvements and alternatives that may will contribute to achieve a better accuracy.

10.2 Future Work

There are a lot of possible improvements and changes to the architecture or the database that could give much better results and some of them explaining below.

10.2.1 Transformers Architecture

When exist variations in between the frames of a video not all the frames could be equally necessary to determine its class. In those things, swinging a self-attention-layer within the sequence model can seemingly yield higher results.
For that the design will amendment to a transformer architecture would be a decent choice, as a result of this can assist on focusing on some specific info
To succeed that the network should decide that regions is a lot of necessary than alternatives and mistreatment for mining the helpful information from them and discard-or fade out- the remainder.
For this scope, the most method is to calculate the scores-higher or lower- counting on the relevance of every feature. (31)

First, should add Self Attention Layers which can build a self-attention mechanism to form global dependencies between inputs and outputs of the layer within the Transformer-based design. D
The self-attention mechanism permits the inputs to act with one another ("self") and figure out to whom they ought to pay additional attention ("attention"). The outputs are aggregates of those interactions and attention ratings. (32)
The basic building blocks of a Transformer are order-independent. Since videos are ordered sequences of frames, the Transformer model must account for the order information via positional encoding, where the positions of the frames in the videos are simply embedded with an Embedding layer. (33)

Position encoding describes the location or position of an entity in a sequence so that each position is assigned a unique representation
Transformers use an intelligent position encoding scheme where each position/index is mapped to a vector. Thus, the output of the position encoding layer is a matrix where each row of the matrix represents an encoded object in the sequence that is summed with its position information. (34)

Factoring outputs into multiple independent spaces, adding residual connections, adding normalization layers - these are all standard architecture patterns that should be used judiciously in any complex model. Together, these bells and whistles make up the Transformer encoder.

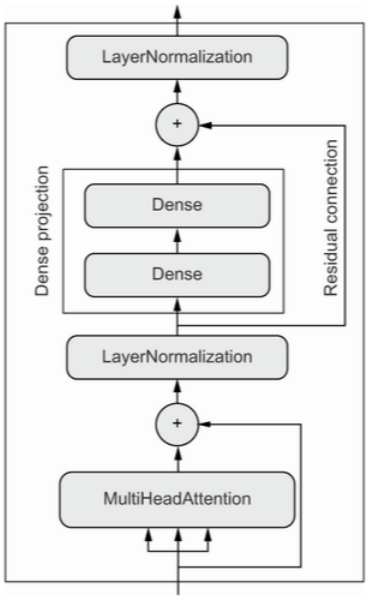


Figure 38 Transformation Model (34)

For sequence data, it's a good idea to use a LayerNormalization layer that normalizes each sequence independently of other sequences in the batch. This layer summarizes the data within each sequence separately for use in creating an image classification model.
A transformer is a sequence-to-sequence model. This means that it is designed to transform one sequence into another. (35)
This is a good practice to rank the value of word order information for classification.

10.2.2 Categorize by the meaning

Another good idea to achieve greater accuracy is to separate the labels of the training videos by their meanings. This way, the network can categorize the videos and separate them depending on the semantics of the words that they represent. After that, the spectrum of the labels will be smaller and the possibility of a prediction will be increased.

10.2.3 Training on a phrase

A language is its phrases and proposals. The next goal for this model is to continue the training on a dataset where the words will be replaced by whole phrases which, of course, has to make sense. This would be more difficult to succeed but for a complete language recognition, has to be done.

Chapter 12

Bibliography-Websites

(1) RoboticsCNN-RNN: A Unified Framework for Multi-label Image Classification

Jiang Wang | Yi Yang | Junhua Mao | Zhiheng Huang |Chang Huang | Wei Xu | Baidu Research
University of California at Los Angles
https://viso.ai/computer-vision/image-classification/

(2)https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4
(3)https://www.sciencedirect.com/science/article/pii/S1877050921000442

(4)https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20CNN%20typically%20has%20three,and%20a%20fully%20connected%20layer.

(5) https://link.springer.com/chapter/10.1007/978-3-030-04780-1_23

(6) https://deeplearningmath.org/sequence-models.html

(7) https://www.techtarget.com/searchenterpriseai/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap#:~:text=RNNs%20are%20better%20suited%20to,more%20on%20this%20point%20below

(8)https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1

(9) https://keras.io/guides/working_with_rnns/

(10) https://www.kdnuggets.com/2018/02/8-neural-network-architectures-machine-learning-researchers-need-learn.html

(11)https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)#:~:text=A%20transformer%20is%20a%20deep,and%20computer%20vision%20(CV)

(12) https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-network#:~:text=What%20is%20a%20Transformer%20Neural,area%20of%20natural%20language%20processing

(13)https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8434597/

(14)https://www.researchgate.net/publication/221292288_Bi-channel_sensor_fusion_for_automatic_sign_language_recognition

(15) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8749583/

(16)https://www.researchgate.net/figure/Hand-gesture-experimental-set-up-with-four-electrodes_fig2_224182799

(17)https://en.wikipedia.org/wiki/CUDA#cite_note-CUDA_intro_-_TomsHardware-1

(18)https://docs.conda.io/en/latest/

(19) https://keras.io/about/

(20) https://en.wikipedia.org/wiki/Keras

(21)https://dxli94.github.io/WLASL/

(22)https://paperswithcode.com/dataset/wlasl

(23) https://cloud.google.com/tpu/docs/inception-v3-advanced

(24) https://www.geeksforgeeks.org/how-to-get-properties-of-python-cv2-videocapture-object/

(25)https://keras.io/examples/vision/video_classification/

(26)https://keras.io/api/layers/recurrent_layers/gru/

(27)https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6

(28)https://keras.io/api/layers/regularization_layers/dropout/

(29)https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/#:~:text=The%20dense%20layer%20is%20a,performs%20a%20matrix%2Dvector%20multiplication

(30)https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c

(31)https://www.researchgate.net/figure/CNN-RNN-architecture_fig1_353254139

(32)https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

(33)https://keras.io/examples/vision/video_transformers/

(34) https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

(35)https://livebook.manning.com/book/deep-learning-with-python-second-edition/chapter-11/274

(36) Hand gesture recognition using a neural network shape fitting technique
E. Stergiopoulou | N. Papamarkos
International Journal of Machine Learning and Computing, Vol. 9, No. 6, December 2019
https://www.sciencedirect.com/science/article/abs/pii/S0952197609000694

(37) Static Sign Language Recognition Using Deep Learning
Lean Karlo S. Tolentino | Ronnie O. Serfa Juan | August C. Thio-ac | Maria Abigail B. Pamahoy | Joni Rose R. Forteza | Xavier Jet O. Garcia
International Journal of Machine Learning and Computing, Vol. 9, No. 6, December 2019

(38) Automated AJCC (7th edition) staging of non-small cell lung cancer (NSCLC) using deep convolutional neural network (CNN) and recurrent neural network (RNN)
Dipanjan Moitra | Rakesh Kr. Mandal
Moitra and Mandal Health Inf Sci Syst (2019)

https://doi.org/10.1007/s13755-019-0077-1

(39) Host Based Intrusion Detection System with Combined CNN/RNN Model
Ashima Chawla | Brian Lee | Sheila Fallon | Paul Jacob Athlone
Institute of Technology, Athlone, Ireland

(40) A Review on Deep Sequential Models for Forecasting Time Series Data
Dozdar Mahdi Ahmed | Masoud Muhammed Hassan | Ramadhan J. Mstafa
Published 03 Jun 2