**Project Paper**

**Automated Lip Reading**

Submitters: Tal Noam, Amitai Zamir, Evgeny Vendrov

Instructor: Lee-Ad Gottlieb

Contents

[Abstract - 4 -](#_Toc53490954)

[Introduction - 5 -](#_Toc53490955)

[Methods - 7 -](#_Toc53490956)

[Architecture - 7 -](#_Toc53490957)

[Structure - 9 -](#_Toc53490958)

[Convolutional layer: - 10 -](#_Toc53490959)

[3D convolution: - 11 -](#_Toc53490960)

[Max pooling: - 12 -](#_Toc53490961)

[VGG blocks: - 13 -](#_Toc53490962)

[time distributed layer: - 13 -](#_Toc53490963)

[GRU: - 14 -](#_Toc53490964)

[bidirectional GRU: - 15 -](#_Toc53490965)

[Fully Connected Layer: - 15 -](#_Toc53490966)

[ReLU: - 16 -](#_Toc53490967)

[Softmax: - 16 -](#_Toc53490968)

[Dataset - 17 -](#_Toc53490969)

[Tools - 17 -](#_Toc53490970)

[Results - 18 -](#_Toc53490971)

[Discussion - 21 -](#_Toc53490972)

[Difficulties - 21 -](#_Toc53490973)

[is there a point to continue studying the subject? - 23 -](#_Toc53490974)

[Can this field help humanity? - 24 -](#_Toc53490975)

[Applicable uses - 25 -](#_Toc53490976)

[Conclusions - 26 -](#_Toc53490977)

# Abstract

Our goal was to develop an algorithm that will receive a voice-less video of a speaking person as input and will output everything the person said as text.

The dataset we use contain 5113 videos of a variety of people, each video is one letter of the English alphabetical.

Our algorithm is based on Deep Learning tactics such as CNN GRU. We used libraries like TensorFlow, Keras, pandas, NumPy, SciPy, OpenCV, etc...

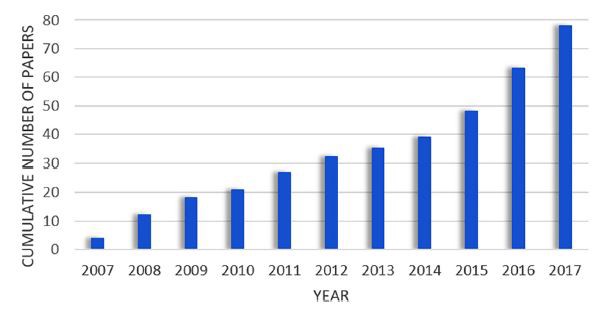
Our research of Existing products in this field revealed that there are papers which try to solve the "lip read" problem with non-deep learning methods, we found methods using CNN to try and find phonemes or viseme on still images, and there are some deep learning products such as "LipNet" which try to achieve the same goal as we do, additionally, we found this paper[1] from about a two years ago which deals with this problem, so this is possible, and there is already done the research to work and to compare our research to.

# Introduction

Our goal is to develop a machine learning algorithm, based on DNN architecture, which will be "trained" to extract text out of a voice-less video of a speaking person, based on the movement of the speaker's lips.

How do we even know it is even possible?

Our first, intuitive, indication that learning how to read lips is possible – is the fact humans do it with great efficiency for centuries. As we researched the topic, we found that with the advancing of Computer Vision and Deep Learning technologies – ALR Machine Learning based systems became more and more popular, this advancing in the DL field made the problem “solvable” as the deterministic algorithm approach yields bad results.  
We found that the cumulative number of papers on ALR systems published between 2007 and 2017, was growing rapidly, as seen below:



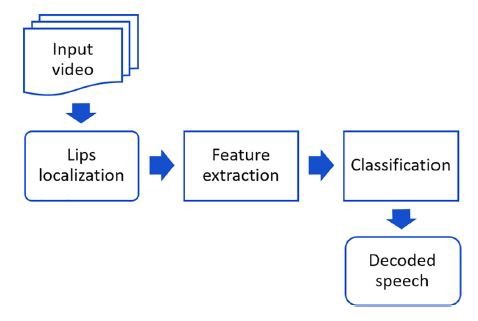
Out of many studies, we found - two relatively recent once (‘16 and ‘17) achieve high accuracy on their datasets and use DNN architectures – which we thought we could use to achieve efficiency. We now have the proof that creating an ALR system using a DNN model is possible.

Our dataset consists of high-quality videos - facial recordings of sentences spoken by various talkers called lombardgrid which we found online. It contains 5113 videos and the transcript of each video.

Trying to solve a problem using Deep Learning methods requires a lot of computing power, that’s why we used AWS Sagemaker, the instance we chose called – ml.p3.2xlarge with a total of 8 CPUs and GPUs, 61 GiB memory, and 100GB local memory.

# Methods

Architecture: This image below demonstrates the idea of the system we're about to develop

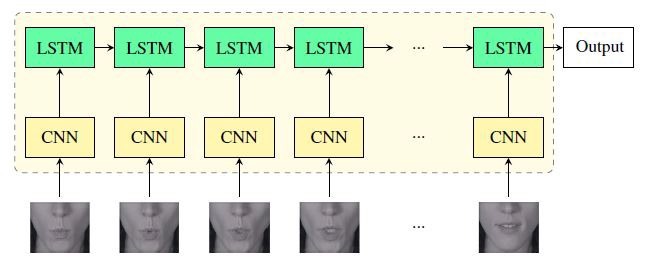


The first two steps as seen in the chart – lips localization and feature extraction are a computer vision problem while classification and outputting text are an NLP problem, so we'll combine two sorts of models, one to deal with videos and pictures and the other to deal with the raw textual data received from the video – outputting it as a correct English sentence.

The whole model will receive a video as input, the first step will be the lips localization and detection, this problem is already solved and researched – so we will use a mixture of open-source tools-based on CNN as the open CV library, to deal with it.

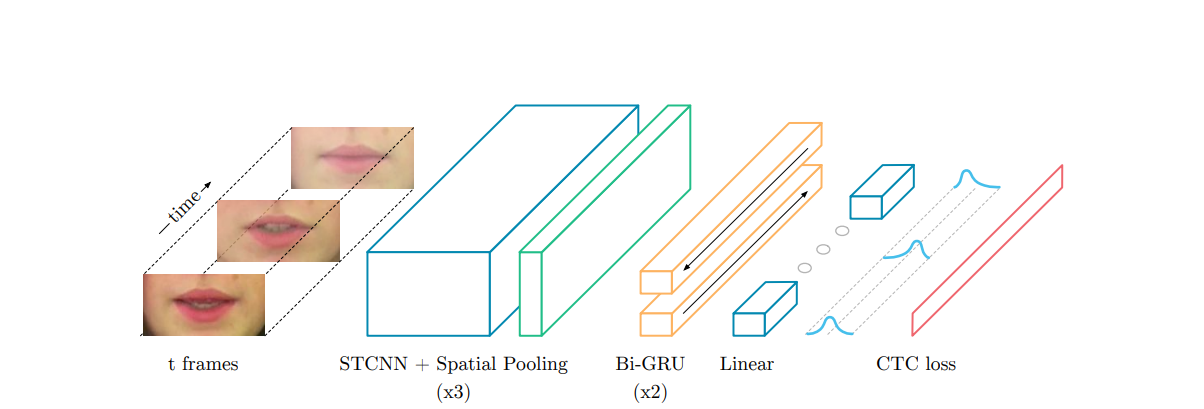
The second step – will be extracting relevant features from the frames of lips we obtained from the video, this is tricky – we don’t know yet which parameters of the speech will be used as features for the NLP component.

The third step - will be taking the output raw-data from the first component as some kind of formatted text data (as CSV), classify it to probably syllables, and then use it as a feature for the second NLP, GRU based component.



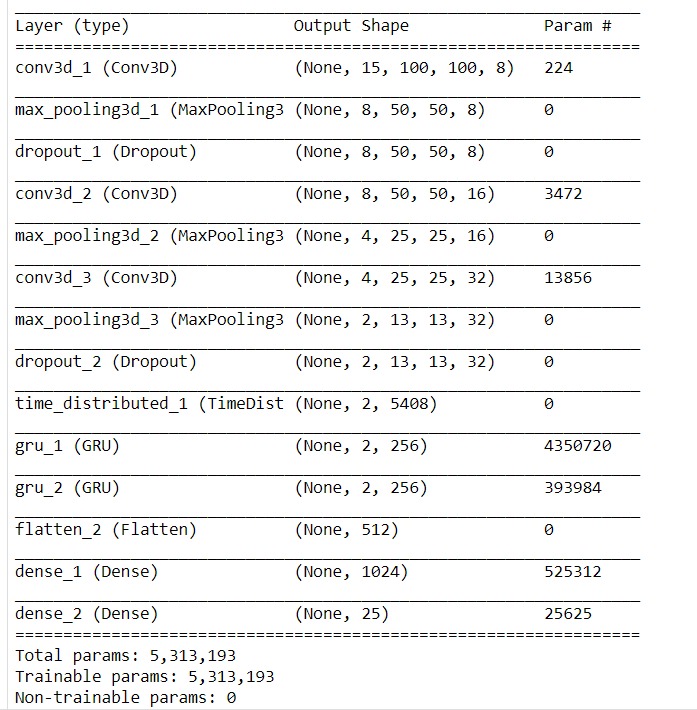
This sort of network matches our dataset as described earlier, it will require a video as input and text as output.

The loss function will be some sort of MSE between what has been said in the video and what the model predicted.



Here above you can see a simulated draw of our model.

Structure: The model structure starts with the CNN class that contains 3 convolution layers include max pulling and drop out after each layer, after time-distributed the output from the CNN used as input into the RNN class that contains 2 GRU layers, then we flatten the matrix of parameters to a vector that process into a couple of hidden layers, and at last cross-entropy loss function. Sum up to over 5.3 million trainable weights!

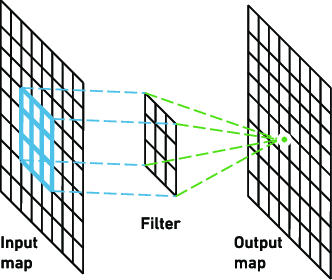


Convolutional layer:

The name “convolutional neural network” indicates that the network employs a mathematical operation called [convolution](https://en.wikipedia.org/wiki/Convolution).

Convolution is a specialized kind of linear operation, Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Illustration of 2D convolution – Input map, in our case, is the input matrix which represents the image, the filter is the sub-matrix which "convolves" values from the source matrix to the output map, which is the "next layer" matrix – i.e. the output of this layer's convolution.

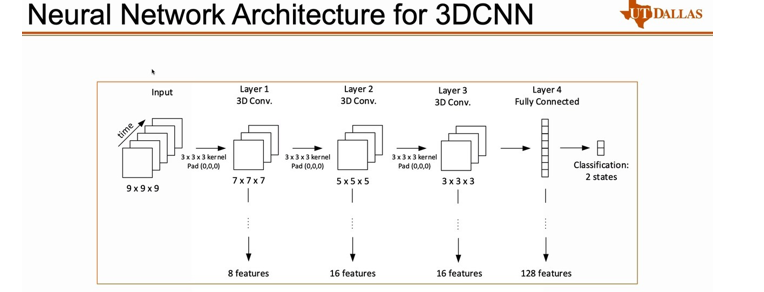


As we try to read lips from a sequence of images, while every sequence has its meaning (i.e. represents a different letter in our case) - every data-point in our model is 15 2D frames, so we need to use 3D convolution.

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery – we use it in our model to locate the lips of the speaker and extract features out of this extracted "lips image".

3D convolution:

Is very similar to 2D convolution, the difference is the input matrix, filter sub-matrix, and output matrix are all 3D, in our case for the first convolution, for example – the input matrix dimension is 15x100x100x3 – 15 is the number of frames represent each letter, 100x100 is the resolution of the image and we got 3 channels as the images are colored (RGB), this way, we convolve on every image group as a single data point.

Illustration of a 3D convolution – please notice that these dimensions aren't equal to the once we work with, in our case, in the first convolutional layer for example - every group of 15 frames is convolved by a group of 32 submatrix's sized – 3x5x5, with the stride of (1, 2, 2), which returns a 13x48x48x32 matrix – we got a total of 3 such convolutions in our model.

Max pooling:

Max pooling is a **sample-based discretization process**, the objective is to down-sample an input representation, reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

This is done to reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

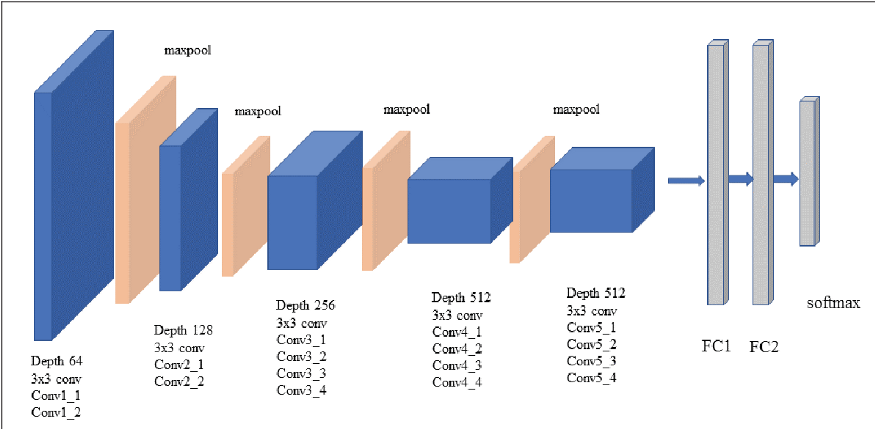
Max pooling is done by applying a max filter to non-overlapping subregions of the initial representation.

Illustration of a 2x2 Max Pooling – the "2x2" represents the size of the pooling sub-matrix.

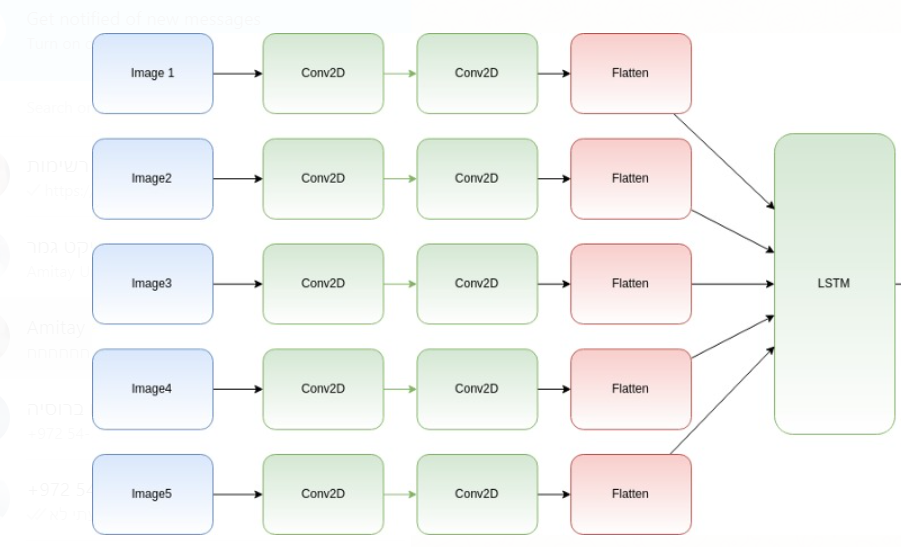
In our case – we work in 3D so the sub-matrix size is 1x2x2 and the stride is 1x2x2 respectively, the size of the output matrix is 13x24x24x32.

VGG blocks:

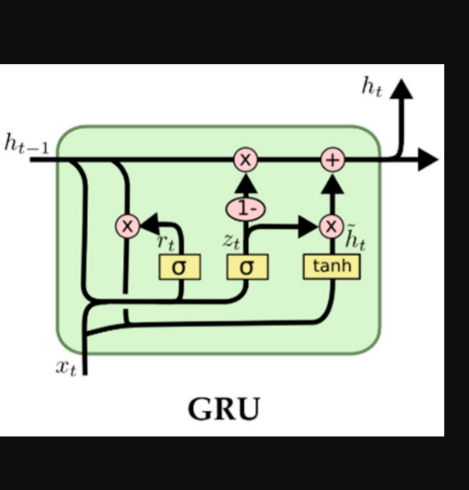
VGG blocks are Convolutional Layers connected to a Max Pooling layer, usually - the amount of the filters grows by powers of 2 –16,32,64,128, etc. VGG blocks are known for being capable of classifying images with large accuracy.

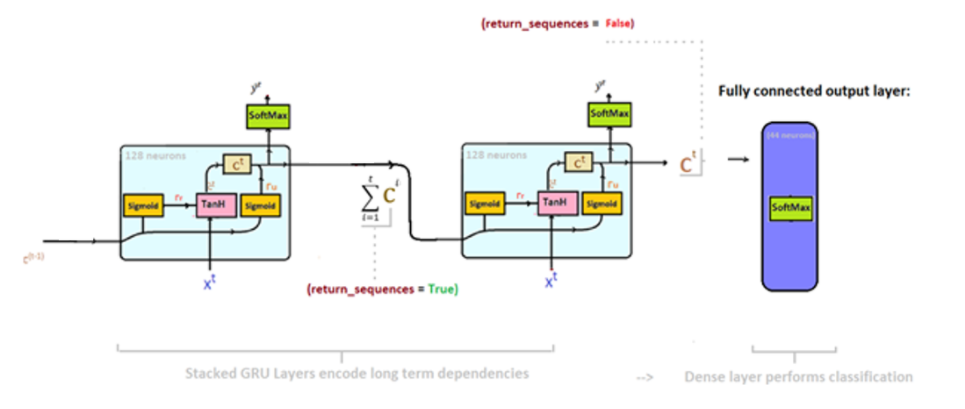
Illustration of a DNN built around 3D VGG blocks –

time distributed layer:

This wrapper allows to apply of a layer to every temporal slice of input, the input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

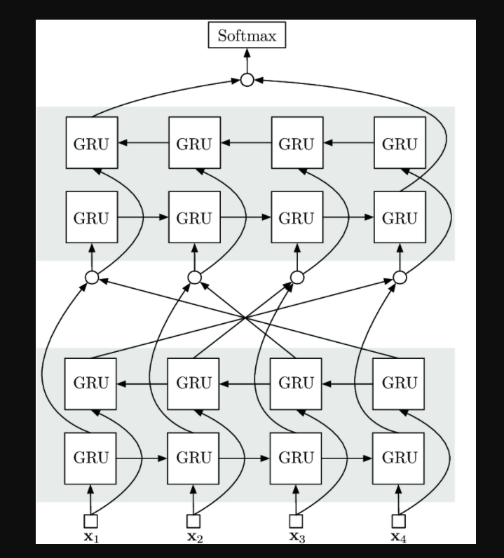
GRU:

Gated recurrent units are a gating mechanism in [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks), The GRU is like a [long short-term memory](https://en.wikipedia.org/wiki/Long_short-term_memory) (LSTM) with a forget gate but has fewer parameters than LSTM, as it lacks an output gate.GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets. Illustration of a one Gated recurrent unit with all its gates and activation functions –

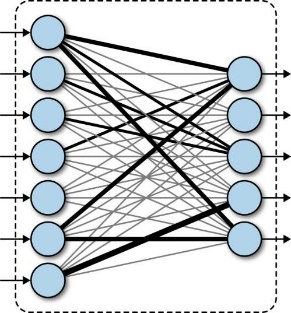
Illustration of GRU net which connects to Softmax (as it is in our model)-

bidirectional GRU:

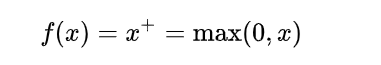
Bidirectional GRU’s are a type of bidirectional recurrent neural network with only the input and forget gates. It allows for the use of information from both previous time steps and later time steps to make predictions about the current state.

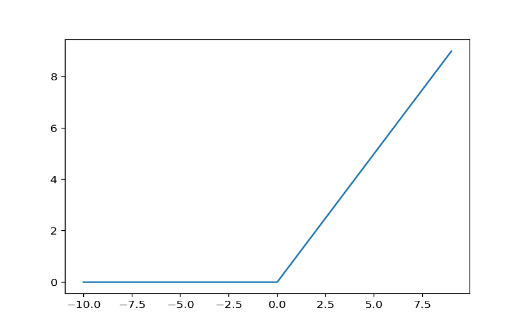


**Fully Connected Layer**:

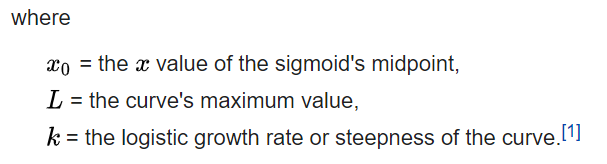
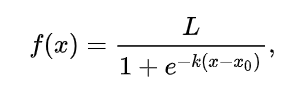
Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network).

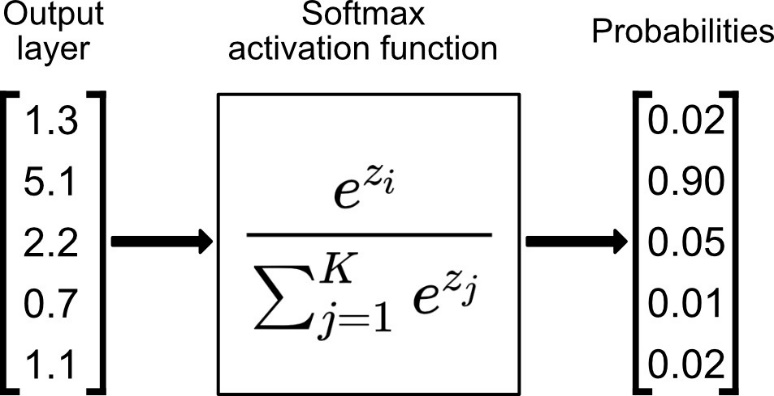
ReLU:

The activation function of a node defines the output of that node given an input or set of inputs. the rectifier is an [activation function](https://en.wikipedia.org/wiki/Activation_function) defined as the positive part of its argument:



Softmax:

A logistic function or logistic curve is a common S-shaped curve ([sigmoid curve](https://en.wikipedia.org/wiki/Sigmoid_function)) with an equation.

The softmax function is a generalization of the [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to multiple dimensions. It is used in [multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression) and is often used as the last [activation function](https://en.wikipedia.org/wiki/Activation_function) of a [neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) to normalize the output of a network to a [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) over predicted output classes, we use it as our output layer activision.

Dataset: the dataset we use contains sentences spoken by various talkers called lombardgrid[2] which we found online. It contains 5113 videos and the transcript of each video. To use the videos in a neural network such as CNN we divided each video into 15 single frames.



Each iteration contains 15 frames of the same person saying a latter of the English alphabetical. The frames go one by one into the CNN and as sequence into the RNN.

Tools: we used a lot of tools in this research such as open-source libraries like TensorFlow, Keras, NumPy, SciPy, OpenCV, etc..

But the tool we have the most use with was AWS Sagemaker. We tons of troubles using this PAAS, our lack of experience this the tool made us sits for hours trying to figure out things, but more on that later. Sagemaker allows us the train the model completely on a outsource server that is a supercomputer with a total of 8 CPUs and GPUs, 61 GiB memory, and 100GB local memory.

The model was written in python, Sagemaker includes an environment service called Jupiter notebook, wich support python 3 and all the libraries we use.

# Results

To run the model, we use AWS Sagemaker, during the training of the model we changed it a few times to improve the training rate and to end up with better accuracy, although there were some improvements, the final result was the same. We tried to fix the model in many ways like add convolution layers or change the GRU, but none have worked. The model did not achieve its goal.

We wanted to compare our results to the Lipnet research results since our model is base on the Lipnet model. The Lipnet model used 75 frames as input, each frame is a mouth image with a resolution of 100X50.

In our model, the input is 15 frames, each frame includes the whole face and not only the mouth at a resolution of 100X100.

The train and validation accuracy during the training was very low: the training accuracy was between 0.02 – 0.1.

The validation accuracy was around 0.0480.

At some point, the accuracy of the train was almost zero and at this point, it never went up. So our model did not manage to learn how to read lips.

We conclude that our model was not "deep enough", unlike Lipnet, we gave our model a "face frames" and tried to make it understand that it needs to be focused on the lips in the frame so we added an extra work, which means we needed a stronger model with more CNN layers to process the frame.

Another conclusion is that our frame resolution was too small. With a face frame of 100X100, there are not enough parameters to learn from, also, the diminution of the frame from 200X200 to 100X100 distort it and make it very difficult to learn so our assumption is that besides of adding more CNN layers we also needed to input frames with higher resolution. The Lipnet research reaches 95% accuracy.

Lipnet research did train their model with a mouth image and prevent the model from the trouble of also learn on what it needs to focus on and only focus on the pattern of the mouth in each frame. In our research, we tried to train the model with face images that include eyes, nose, hair, and even a small background, so our model needed to learn not only the mouth pattern, but it also needed to learn to focus on the mouth. Also, the Lipnet researches trained their model to read sentences, we trained our to read single words from the English alphabetical, and although we supply our model with bigger frames(100X100 vs 100X50) they manage to get an accuracy of 95% which show how the focus on the mouth is much more useful than providing higher resolution frames.

Another attempt we tried was to train the model on two words, 'a' and 'b', and hope to accomplish learning in the process.

After the earning process, we tried to test the model, the results were around 50% accuracy, which means the model fail to learn during the train.

# Discussion

Let's go back a little and remember the question we were trying to solve in this research, the question was, can we teach a machine how to read lips at least as good as people can.

To answer this question we need to include every aspect of this research, our architecture, results, approach, and most importantly past research.

The reason we chose this project was our common interest in Deep Learning. We wanted to take a project that we feel passionate about, and in specific ALR was very intriguing to us. To be honest this project caused us lots of troubles, from the beginning till the very end.

Difficulties**:** Eleven months ago we started to work on the ALR project, we found out very soon that the most difficult thing about this project was to find a good solid dataset, we were looking for a very specific dataset, this dataset needed to include videos of people speaking to a camera and saying single words, also we needed the transcript of each video with the timing of each word, we came across e few datasets that were close to our need, but we kept on looking for weeks for the perfect dataset. Finally, we found the lombardgrid dataset, that was created for the same purpose we needed. One of the most challenging problems we had with the data was to edit each video into frames, We wrote a script in python that checks in the alignment of the video when does the person say the word we need, go to this specific time in the video and take the next 15 frames and save it as images.

Another difficulty we encountered was the AWS platforms. As a student, we never had the chance to work with PAAS, and learning to use AWS was one of the most time-consuming things in this project.

We had to use two services on AWS: S3 and Sagemaker.

S3 is a storage service provided by Amazon to help the AWS users to upload, save, and use data in a "local" way while using other services like machine learning, etc.

Sagemaker is a service that provides users the ability to practice machine learning methods and algorithms. The service let you choose an instance according to the needs of the user, the more powerful the instance, the higher its price. Working with Sagemaker raised many problems along the way, each time we fixed one problem and want to move forward to the next task we found another problem. One of the greatest problems we had was the machine stops after 12 hours of work and also the console output with the information about the run was getting all mixed out. We solve this problem by connecting every few hours into the instance and make sure it works, and we added to the script a separate doc that collects all the data we need from the console.

Another problem we had was to choose an instance, we didn’t know what we need for our machine, we just knew we need something strong. We started to read and learn about the application of the instance and more suitable for our purposes. We found that \*\*ENTER NAME\*\* had the power we needed in CPU, GPU, memory, and RAM.

is there a point to continue studying the subject?

Yes, although there was some progress in this subject, there is a lot more room to improve, the Lipnet research ended with great results but its vocabulary was very small and basic, so the next stage will be to train a model to understand more complex words and sentences or even sentences with meaning and not only colors and numbers or in other words, make the model unsupervised so that it will be more flexible and know how to handle with complex sentences. so if we continue to research we might find a way to build a model that not only knows how to read the lips but also know how the whole sentence should be.

Can this field help humanity?

if we manage to train a model that cal read lips and understand whole sentences, this could make the life of deaf people much easier furthermore, security cameras could tell the man who watches it what the people in the camera talking about and we will train a model on different languages, it could translate what someone saying in a different language so someone who doesn't speak that language will know what the other man saying, it could help people from other countries to communicate with each other.

**but what about boundaries? do people want someone they don't know spying on everything they say? where does it end?**

what if some government will decide to use hidden cameras in public places and start to spy on every word of every citizen in the disguise of public security and counter-terrorism tool.

the people that explore this area should ask themselves where is the boundary lies, and what purpose will it serves.

Applicable uses**:** Companies could create software that can help deff people to understand what other people saying. In the future, there might be a build software on our tv that could read the lips of the people on the tv and translate it to the watchers, an application like that would be useful not only on tv but also in video stream websites like youtube. Another idea is to build some AI that in addition to reading lips could make a conversation with a human by understanding what he says and how to respond

# Conclusions

Throughout the last year, we have been involved in this project, we have occasionally come to conclusions about the field we have chosen here.

One of the main conclusions we have reached is that in a project like that you need a lot of knowledge in different and extensive areas, most of the problems in the project we encountered we had to go deeper and research solutions online to solve them. These problems could be solved with knowledge and experience in the field, things we did not have.

Another conclusion we drew, from a technical point of view, was that the frames we turned into images for the machine were not good enough. We used 100X100 face images while other studies used 100X50 lip images. We think we had to allow larger images for the machine to learn, but we ran into memory problems (each 100X100 image opens to a matrix much larger than it, double 15 frames for each video) so we could not enlarge the images. Our conclusion was therefore that the use of lip images only is absolutely necessary for learning.

Finally, we are glad that we got to work on a project of this kind. We learned a lot of useful and interesting things in the field of machine learning and in particular in deep learning. We got into dealing with technologies that we would not be sure to use in another project like AWS and Python learning libraries. We hope to continue dealing with the field in the future in our careers as well. Thanks for reading. Tal Amitai and Evgeny.

# References:

[1] LipNet <https://arxiv.org/pdf/1809.02108v2.pdf>

[2] lombardgrid <http://spandh.dcs.shef.ac.uk/avlombard/>