UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

Advanced Deep Learning Techniques for Lip Reading

Emilian-Claudiu Mănescu

Thesis advisor:

Șl. dr. ing. Dumitru-Clementin Cercel

BUCHAREST

2022

CONTENTS

SINOPSIS

Cititul pe buze, cunoscut și ca recunoașterea vizuală a vorbirii, este un subset al Vederii Artificiale, care are scopul de a analiza și prezice cuvinte vorbite din videouri, fără a se folosi de informații auditive. Domeniul a cunoscut o creștere semnificativă în ultimii ani, în principal datorită dezvoltării hardware-ului performant și a inovațiilor din cadrul Vederii Artificiale. În această lucrare, voi prezenta tehnici și arhitecturi avansate pentru cititul pe buze, care au fost testate pe setul de date în limba română LRRo, care este disponibil public. În primul rând, am implementat și raportat rezultatele antrenării modelelor de ultimă generație, folosind metode diferite pentru a captura corelațiile temporale ale mișcărilor buzelor. În al doilea rând, am folosit distilarea cunoașterii pentru a obține un model mai puțin intensiv computațional, care să obțină rezultate comparabile cu modelele de ultimă generație. În al treilea rând, am utilizat setul de date recent publicat GLips, în limba germană, pentru a îmbunătăți semnificativ performanța modelelor pe setul de date in limba română, folosind antrenare adversarială si adaptarea domeniului.

ABSTRACT

Lip reading, or visual speech recognition, is a subset of Computer Vision, which aims to analyze and predict spoken words from videos, without the aid of auditory information. The field has seen a great upsurge in recent years, particularly because of the development of high-performance hardware and the innovations in Computer Vision. In this paper, I present advanced techniques and architectures for lip reading, which are tested on the publicly available Lip Reading Romanian dataset (LRRo). Firstly, I implemented and reported the results of training the current state-of-the-art models on this dataset, using different methods to capture the temporal correlation of lip movements. Secondly, I use knowledge distillation to obtain a less computationally-intensive model, which obtains comparable results with fewer operations. Thirdly, I adopt the recently published German-language dataset (GLips) to significantly improve the models' performance on the Romanian dataset, using domain-adaptation adversarial training.

1 INTRODUCTION

1.1 Context

Lip reading represents the task of understanding and classifying the words spoken by a person, using only the visual cues of the speaker's lip movements. It has been gaining traction over the last few years, principally due to increased computational resources and its inherent usefulness in real-world applications: surveillance, assisting hearing impaired persons, biometric identification. Human lip reading accuracy is notoriously low, therefore there has been increased interest in using deep learning techniques to surpass it.

1.2 Problem

Lip reading has been extensively researched in recent years. However, most of the models trained have been tested and optimized on only two large-scale datasets, LRW (English-langauge) and LRW-1000 (Chinese-language). This is useful for research purposes, but it does not demonstrate the models' ability to learn complex features on smaller datasets, for less common languages such as Romanian. This naturally raises the question of using the existing data to get higher accuracy, even if it comes from a different language. Moreover, these models are CPU and memory intensive, since they have a large number of parameters, meaning that they cannot be easily deployed on devices with limited computational resources such as embedded systems or mobile devices.

1.3 Objectives

The objective of this paper is to provide a comprehensive overview of the best performing deep learning architectures and techniques for the task of lip reading. I will also question the need for complex and computationally-intensive models by using knowledge distillation and testing the efficiency of smaller architectures which can easily be deployed on mobile devices. Moreover, I will determine if the general accuracy of a lip reading model can be improved by leveraging unlabeled samples from an unrelated dataset, therefore paving the way towards language-invariant lip reading, which is particularly important for underrepresented languages such as Romanian. This is also relevant since it reduces the need for large-scale datasets, which are very expensive to produce since they require a lot of preprocessing, manual annotation and proof-checking. Moreover, this paper aims to provide a new baseline performance for the

Romanian-language dataset Wild LRRo, for which, to the best of my knowledge, few models have been tested.

1.4 Structure

The following chapter will present background information needed to understand the solutions proposed and to form an intuition about the lip reading task. Firstly, theoretical notions about deep learning in general will be presented, ranging from vanilla artificial neural networks to recurrent neural networks and knowledge distillation, which are the basis of the architectures that I tested in this paper. The lip reading task is also addressed, along with its motivations and related work, highlighting its evolution across the years, especially since the resurgence of deep learning techniques. Moreover, the frameworks and tools used in the implementation are listed and detailed. Lastly, the two main datasets used in this paper are described comprehensively, namely the Romanian language LRRo and the German language GLips, both of which have been proposed recently and are generally underrepresented in the field.

The third chapter provides a comprehensive description of the architectures used, ranging from the standard deep learning models, which have been inspired from related literature, to the two types of knowledge distillation techniques applied, and lastly, to the models trained adversarially using domain adaptation. Emphasis is also placed on the several optimization and regularization methods used to improve accuracy and generalization on unseen training samples.

The fourth chapter starts by detailing the context and environment in which the solutions proposed were implemented and tested, as well as a brief overview of the metrics used to assess performance. Moreover, it presents an extensive analysis of the models trained, along with a comparison of the best performing architectures and hyperparameter configurations. Furthermore, an analysis of the Romanian dataset is provided, in order to have a clear perspective of the results obtained.

The fifth and last section consists of general conclusions taken from the experiments conducted, with a last paragraph describing possible future improvements.

2 BACKGROUND

This chapter will present the prerequisite notions needed for understanding the methods and tools used in developing the proposed solutions, while also explaining the motivations and previous works related to the lip reading task.

2.1 Theoretical Background

2.1.1 Deep Learning

Deep Learning [1] is a subfield of Machine Learning and Artificial Intelligence, which loosely draws influence from human brain anatomy to model abstract and complex tasks (but certainly failing to match its capabilities). It has enabled multiple breakthroughs in speech recognition, image classification, event detection, biometric identification, text analysis and machine translation etc.

Deep Learning algorithms can be classified into the following categories:

- supervised learning - the task of learning a mapping of inputs to labeled outputs. The model is directed (using complex mathematical operations such as function differentiation and gradient descent) towards a state in which its parameters approximately describe the mapping, while attempting to generalize on unseen data pairs. Therefore, learning underlying patterns in the data is favoured over simply memorizing input-output pairs. It can be further subclassed into two types of tasks: regression (predicting continuous data) and classification (separating the samples into a finite number of classes, based on their attributes). One major disadvantage of this learning paradigm is the need for gathering large amounts of training data, a costly and time-consuming process, which often involves expert intervention.
- unsupervised learning - the task of learning hidden patterns from unlabeled data. As opposed to supervised tasks, it can't be used for classification or regression problems, but rather for understanding the structure of the dataset. It is mainly used for clustering (grouping samples based on a similarity function) and dimensionality reduction (projecting complex data on a smaller dimensional plane, either for visualization or for reducing the amount of features in the dataset, while maintaining the same variance).
- semi-supervised learning - a combination of the two previous paradigms. It describes the models that use a large amount of unlabeled data and only a few labeled samples during training. It is mainly used for situations in which data annotation is costly and

infeasible.

- reinforcement learning [2] - describes those learning algorithms in which an agent decides sequentially what action to take, based on a weighted sum of immediate and predicted future rewards, which it tries to maximize. It is mainly used in robotics, computer games, traffic control systems and advertising.

2.1.2 Artificial Neural Networks

Artificial Neural Networks are computing systems modelling complex non-linear functions. They are designed to roughly mimic the neurons and synapses in the human brain. They are constructed as layers stacked on top of each other, each one receiving its inputs from the previous layer, executing complex matrix multiplications on them before sending them on to the next layer. Generally, the capacity of a neural network to learn complex mappings rests upon the number of hidden layers, the number of neurons in each layer, as well as upon the activation functions, which apply non-linearity to the results of the multiplications.

The process of making a neural network learn involves two phases, which alternate during training:

- forward pass - in which the input of the neural network is passed towards the output layer in order to compute the result associated with that input (continuous value for regression or class prediction for classification tasks).
- backward pass (also known as backpropagation) [3] - in which the gradients of the loss function (which measures the error between the predictions of the ANN and the real values) with respect to the neural network weights are being propagated from the output layer towards the input layer (hence the name backpropagation)

The gradients computed in the backward pass will be used to update the values of the network parameters using an optimization algorithm, which are improvements over the standard gradient descent method. Generally, the purpose of deep learning optimizers is to modify the weights of the NN so as to reach a point of global minimum for the loss function (so a point where the neural network outputs and the true labels are as similar as possible).

In standard gradient descent [4], the update is done as follows:

$$W_{k+1} = W_k - \alpha * \frac{\partial J}{\partial w} \tag{1}$$

where W_k and W_{k+1} are the layer weights at time steps k and k+1, α is the learning rate, which determines the size of the gradient descent step at each iteration, and $\frac{\partial J}{\partial w}$ are the gradients computed during backpropagation.

2.1.3 Convolutional Neural Networks

Convolutional Neural Networks are the basis of Computer Vision. They have been first used in a modern deep learning model by LeCun et al. [5], when they used their proposed LeNet architecture on the MNIST database of handwritten digits.

Although vanilla neural networks have been proven to be very powerful in understanding the underlying patterns in data, they are not efficient when it comes to processing images, since these would first need to be flattened to a 1-dimensional array and then fed into the neural network. This exponentially increases the number of parameters and discards the spatial information of the image. For example, an RGB image of size $228x228x3$ entails an input layer of 155952 units.

Convolutional neural networks improve upon these drawbacks by using convolutional kernels, which act as feature detectors that slide across the image, capturing relevant spatial information in the input image. Figure 1, which is taken from [6], presents the numerical operations performed by a 3x3 kernel on a 5x5 image.

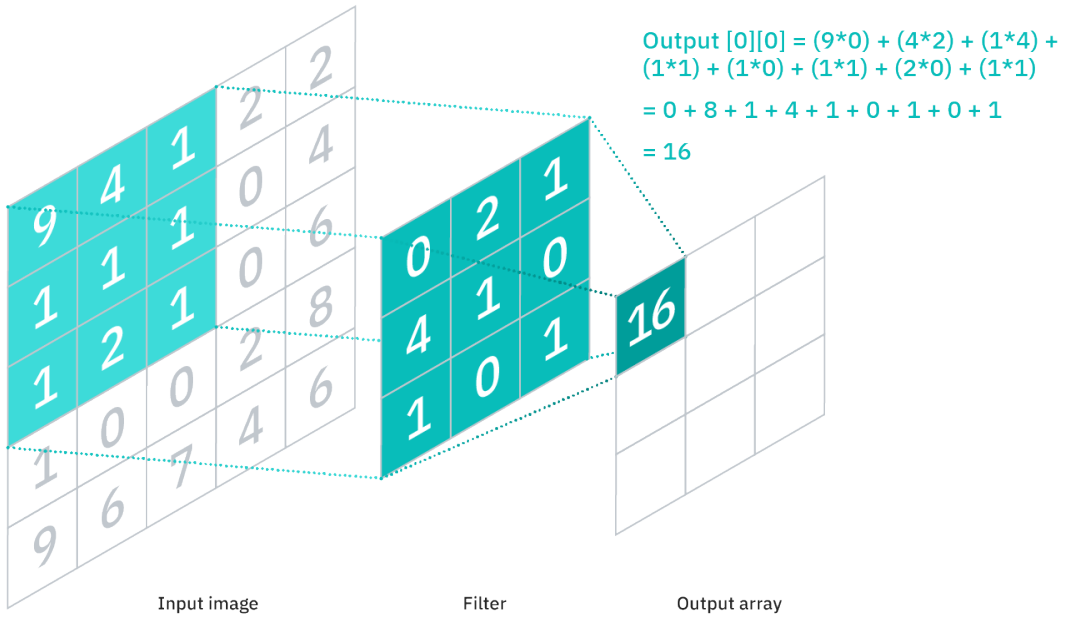


Figure 1: Convolutional filters

Convolutional neural networks are generally constructed as sequences of convolutional layers (made of several kernels which recognize different patterns in the input image), followed by a nonlinear function (typically ReLu) and a subsampling layer (max pooling or average pooling), which reduces the spatial size of the representation (and therefore also the number of paramters to learn). The weights learned during training are the parameters of the kernels in each convolutional layer. A batch normalization layer [7] can also be introduced between the activation function and pooling layer to ensure that the input distributions of every layer are similar, similar to the feature normalization performed on the dataset before training the

neural network.

It has been, however, observed that the performance of CNNs degrades when too many layers are used. This is because of the so-called *vanishing gradient* problem [8], in which the gradients computed during backpropagation easily approach 0, due to the multiple multiplications in the chain rule of differentiation. To solve this problem and enable the full potential of CNNs, He et al. [9] introduced the ResNet, which uses skip connections to ensure gradient stability by adding an identity mapping which passes information to future layers unchanged. Skip connections avoid vanishing gradients by ensuring that, during backpropagation, gradients can move through fewer layers, therefore guaranteeing that fewer multiplications are performed. Since the connection is simply an identity function (or just passes through a layer which ensures that both vectors have the same size), the gradient remains virtually unchanged. They are also particularly useful since they provide a way for the neural network to decide which layers are useful and which are not, because a layer can simply pass the input through via the identity function. Figure 1 (taken from [9]), shows a basic residual block used in a ResNet.

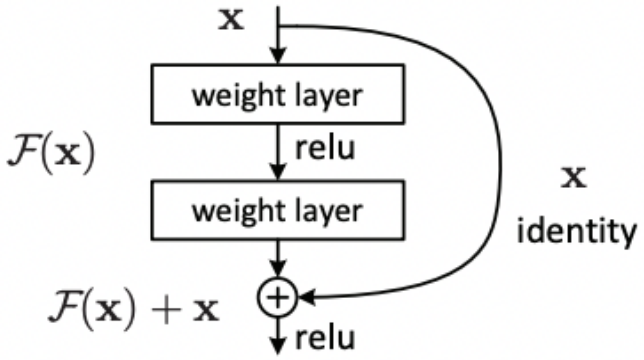


Figure 2: ResBlock

2.1.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are specialized neural networks capable of dealing with and understanding sequential data. Therefore, they are mainly used in natural language processing tasks, such as text summarization, machine translation and conversational agents, but they have also been successfully applied in face detection, optical character recognition, video analysis and even music composition.

The main innovation of RNNs over vanilla neural networks is that they can remember information from previous inputs because of an in-built memory system, which allows them to gain insights based on both the current input and the information learned from previous data. Namely, an RNN computes, at each time step, both an output (the prediction at that time

step) and a hidden state (which acts as a memory retaininng useful information from the past):

$$y_t = f(W_y * h_t + b_y) \tag{2}$$

$$h_t = g(W_x * x_t + W_h * h_{t-1} + b_h) \tag{3}$$

where f and g are activation functions, W_y, W_x and W_h are the weights associated with the output, input and hidden state, respectively, and b_y and b_h are biases.

This architecture also suffers from the vanishing gradient problem because of Backpropagation through Time, which unfolds the neural network along the time axis. To solve this, two RNN architectures have been proposed:

- Long short-term memory (LSTM) [10], which add complex gates and a vector named cell state to the standard RNN. The cell state stores the relevant information from the previous time steps. It is modified by the three cell gates of the LSTM network: the forget gate - which decides which information should be discarded and which should be passed on in time; the input gate - which updates the cell state by deciding the values that will be updated; the output gate - which computes the next hidden state of the network, based on the newly calculated cell state, the input at the current time step and the previous hidden state.
- Gated recurrent unit (GRU) [11], which attempt to solve the vanishing gradient issue by using only two gates (as opposed to the LSTM). The update gate decides which information to keep and use in future iterations. The reset gate establishes how much of the information is no longer needed and should be forgotten. Namely, the numerical operations performed by the GRU unit are as follows:

  $$z_t = \sigma(W_z * x_t + U_z * h_{t-1}) \tag{4}$$

  $$r_t = \sigma(W_r * x_t + U_r * h_{t-1}) \tag{5}$$

  where z_t and r_t are the update and reset gates, respectively, h_{t-1} is the hidden state at the previous time step, and σ is the sigmoid function.

  The reset gate and the input will be used to create the candidate hidden state, h'_t, which contains the relevant information from the past:

  $$h'_t = tanh(W * x_t + U * (r \ \odot h_{t-1})) \tag{6}$$

  The update gate will be used as a weight to compute the hidden state at the current time step, h_t:

  $$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t \tag{7}$$

2.1.5 Knowledge Distillation

Distillation refers to the process of transferring knowledge from a complex model, which can't be used in a real environment without consuming a lot of resources, to a smaller, less

computationally intensive one, while maintaining a similar performance. It can therefore be considered a model compression algorithm. It has been proposed by Hinton et al. [12], who suggested that a smaller model could be trained to predict the probabilities of the larger one, modified to represent a softer, less strict distribution (named "soft" targets). Namely, if z_i are the logits of the teacher model, the soft targets will be computed as follows:

$$q_i = \sigma(z_i;\ T = T) = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{8}$$

where T is a hyperparameter named temperature which ensures that probabilities are not given absolute values (0 or 1) and the model can see which classes are more probable than others, therefore distilling more information into the student model. It can be observed that for $T = 1$, the probabilities computed correspond to the softmax function.

In practice, however, better performance is obtained when optimizing for two loss functions simultaneously, where the first one is the cross-entropy loss between the soft targets of the teacher and soft predictions of the student, and the second one is the cross-entropy loss between the hard predictions of the student and true labels of the training data:

$$L_{KD} = \alpha * L(\sigma(z_t;\ T = \tau), \sigma(z_t;\ T = \tau)) + \beta * L(y_{true}, \sigma(z_s;\ T = 1)) \tag{9}$$

where z_s and z_t are the logits of the student and teacher models, respectively, while α and β are hyperparameters which assign weights to the two losses. The cross-entropy loss is added to ensure that the student model is able to learn more complex representations than the teacher. Higher values for the temperature hyperparameter τ yield better results when the discrepancy between the teacher and student is smaller, since the latter can more easily understand the patterns in the probability distributions, while lower values are usually recommended when the difference between the two models is more significant.

2.2 Lip Reading

Lip reading, also known as visual speech recognition (VSR), is the task of predicting the spoken words/sentences in a video, using only the visual cues of the person's lip movements and discarding the auditory information.

It is considered a difficult problem to solve, because of several reasons:

- datasets are very expensive to produce, since they require a lot of preprocessing, manual annotation and precise alignment between the video and the utterances.
- its inherent difficulty for humans. Hilder et al. [13] observed that human accuracy on monosyllabic words is between 20% and 50%. Machine accuracy is expected to surpass this threshold, but it can't compete with audio speech recognition, for which human accuracy, even in noisy surroundings, exceeds 60% [14].

- the discrepancy between the number of phonemes (smallest unit of sound in language) and visemes (visual equivalent of phonemes), which have a many-to-one mapping. This is caused by the absence of information coming from the tongue and throat, which influences the sound being produced. For example, the sounds /b, /p and /m have the same viseme.
- lip movement information is influenced by pose, lightning conditions and even the emotional state of the speaker.
- absence of context

  Generally, a lip reading task has four steps:

  - input preparation - in which the steps of face detection, lip localization and extracting the region of interest (RoI) are performed.
  - spatial feature extraction (also known as frontend) - typically a type of convolutional neural network which maps the movements and shapes of the speaker's lips to high-level features
  - sequential feature extraction (also known as backend) - which models the sequential correlation between the spatial features extracted
  - classification module - which transforms the features obtained in the previous step into class probabilities.

2.2.1 Motivation

Some notable uses of lip reading include:

- assisting deaf or aging people in understanding speech
- speech prediction in noisy environments, when visual cues from lips could complement the defective audio information
- biometric recognition
- dubbing silent films or assisting people in creating subtitles for videos/films

2.2.2 Related Work

The standard lip reading task was mainly based, before the advent of deep learning techniques, on hand-engineered features, obtained for example using Hidden Markov Models [15]. However, recently, several neural network architectures have been proposed. In their paper introducing the Lip Reading in the Wild dataset, Chung et al. [16] built on top of the VGG-M network and proposed four novel architectures which incorporate early fusion (combining the 25 frames of the video in a single representation) and multiple towers (each frame is passed to a separate copy of the same sequence of convolutional and pooling layers) combined with 3D or 2D convolutional frontends. The same authors proposed SyncNet [17], which detects and corrects the audio-video synchronization error and uses the resulting features to train a LSTM

network on the lip reading dataset OuluVS2 [18]. Assael et al. introduced in the same year LipNet [19], which is the first sentence-level lipreading model. It used layers of spatio-temporal convolutions followed by two bidirectional GRUs and was trained using CTC loss to surpass professional human lipreaders on the GRID dataset [20]. However, it was soon discovered that more complex architectures should be used in order to obtain significant results. Therefore, Stafylakis et al. [21] proposed a network containing a 3D convolution as a spatio-temporal frontend, followed by a 34-layer ResNet and a bidirectional LSTM and improved the state-of-the-art accuracy at the time by almost 7%. Also, a comparison of frontend architectures has been made by Yang et al. [22], when they introduced the LRW-1000 dataset and showed that a combination of 3D and 2D convolutions has the best performance on it. A comprehensive study of the most efficient lip reading training strategies was made by Feng et al. [23] and tested on the same dataset. More specifically, they replaced the standard ResBlocks with Squeeze-and-Excitation blocks [24] and used word boundaries and label smoothing to obtain the current state-of-the-art performance on this complex dataset.

There have been several attempts at distilling knowledge in lip reading models. Shrivastava et al. [25] introduced MobiVSR, an end-to-end neural network designed to be deployed on mobile devices. It used a frontend consisting of two depthwise-separable layers, a stack of LipRes blocks (similar to Residual blocks but using separable convolutions) and a temporal convolutional neural network as backend. Zhao et al. [26] and Afouras et al.[27] realized that auditory and visual information complement each other and leveraged Audio Speech Recognition models to distill cross-modal knowledge to lip readers. Ma et al. [28] raised the accuracy on LRW and LRW-1000 by using self-distillation, in which both the teacher and the student models have the same architecture. They also discovered that the standard distillation process is more efficient when the discrepancy between the two models is smaller, so they employed distillation in generations, using increasingly smaller models when transferring knowledge.

Domain adaptation is currently underrepresented in the lip reading field. Jitaru et al. [29] proposed a transfer-learning approach by creating a multilingual dataset containing samples from Romanian, English and Chinese with the intent of verifying if the models could learn language-independent representations. There have been, however, attempts at using language as a discrimnator. For example, Abdelwahab et al. [30] used unlabeled data from a related dataset to improve acoustic emotion recognition, by employing Domain Adversarial Neural Networks.

2.3 Tools and Frameworks

2.3.1 NumPy

NumPy[1] is a fundamental open-source Python library used for advanced and efficient numerical operations. It provides high-performance data structures (n-dimensional arrays, also known as ndarrays) that make working with numerical data easy and intuitive. They improve upon the standard Python lists, which are designed for general use and can contain different data types, by offering vectorized operations on homogeneous data, which improve both memory and time consumption.

2.3.2 Matplotlib

Matplotlib[2] is a cross-platform Python library, tightly integrated with Numpy, which provides various tools for data visualization, including line plots, scatter plots, histograms, boxplots and pie charts. It is highly customizable, offering users the possibility to design the graphs to their liking, by modifying the axes labels and plot markers, changing the color themes and adding legends.

2.3.3 PyTorch

PyTorch[3] is one of the most popular Python frameworks for building, designing and training complex Deep Learning models. It provides features for developing models as efficiently as possible, and is specifically suited for programs that use both CPUs and GPUs, making migration between the two easier and more straightforward. PyTorch also offers a package specialized in distributed training, which provides users an effortless method of running models on multiple GPUs and even multiple machines, with minimal code changes.

The main features of PyTorch are:

- Tensors - the PyTorch equivalents of NumPy arrays, the major difference being that they can be easily moved to the GPU with a simple method call
- Autograd - module which facilitates efficient gradient computation at runtime. Autograd is the engine that makes PyTorch so well suited for deep learning tasks, since it automates the gradient descent algorithm (which is used to update the model weights during training)
- nn - module for effortlessly building deep learning models. Using the nn module, the user can create the basic blocks of neural networks using predefined classes which abstract

[1] https://numpy.org
[2] https://matplotlib.org
[3] https://pytorch.org

the underlying operations (for both the forward and the backward pass). It is also easy to build custom nn modules to abstract more complex architectures/algorithms.
- Datasets and Dataloaders - primitives which simplify data loading and separation into batches, while minimizing memory consumption.

2.4 Datasets

2.4.1 LRRo

LRRo, proposed by Jitaru et al. [31] from University Politehnica of Bucharest, is a lip reading word-level dataset containing videos and target labels for the Romanian language. It is currently the only Romanian dataset for VSR.

LRRo is split into two distinct datasets, based on the nature of the videos they contain:

- Lab LRRo - with videos recorded in a controlled environment, from 19 different speakers (with or without speech impediments) and a vocabulary of 48 words. The lightning conditions and recording angles are similar for all videos (cameras were placed in a frontal position and 30° to the left of the speaker's head, to add more diversity).
- Wild LRRo - containing videos retrieved from a variety of publicly available recordings (Romanian TV Shows, Ted talks, TV news programmes), with different resolutions, lightning conditions and variable camera angles. The data obtained, albeit smaller in size, contains the utterances of 35 speakers and a vocabulary of 21 words (which are non-uniformly represented, as opposed to the lab-controlled dataset).

Both are divided into three subgroups (train, validation and test set). For Wild LRRo, they are structured as follows: 848 train, 120 validation and 121 test samples, while for Lab LRRo: 6505 train, 860 validation and 815 test samples. Videos are provided as a sequence of 29 *jpg* images, which have been preprocessed and cropped around the speaker's mouth. Examples of frames from the Wild LRRo dataset can be seen in figure 3.

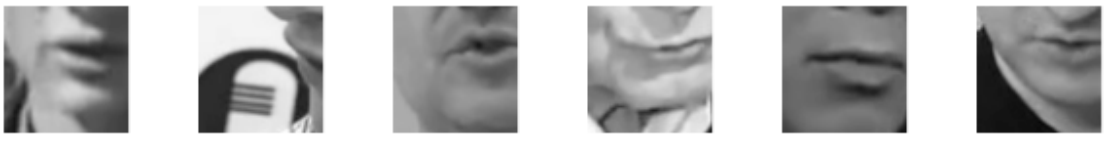


Figure 3: Frames from the Wild LRRo dataset.

2.4.2 GLips

Figure 4: Frames from the GLips dataset.

GLips, proposed by Schwiebert et al. [32], is a German word-level dataset for lip reading. It consists of more than 250,000 videos of 100 speakers from the Hessian Parliament, recorded in an uncontrolled environment (therefore they have variable poses, lightning conditions and emotional states of speakers). It draws its inspiration from the English dataset, Lip Reading in the Wild (LRW) [16], videos from both datasets having the same resolution, length (1.16s), framerates and formats, and even the same number of target labels (500). Word lengths vary from 4 to 18 characters. Videos are provided as *mp4* files and have not been cropped or preprocessed beforehand. Original audio files are also provided. Note that this dataset may contain erroneous data, since subtitles from the Youtube channel of the Hessian parliament were used, which are not 100% reproductions of the speaker's words. Examples of frames from the GLips dataset can be seen in figure 4.

3 PROPOSED SOLUTION

3.1 Standard Lip Reading Models

For the standard lip reading models, I draw inspiration from the papers published by Feng et al. [23] and Martinez et al. [33], who have proposed innovative network architectures which have achieved state-of-the-art performance on the LRW dataset.

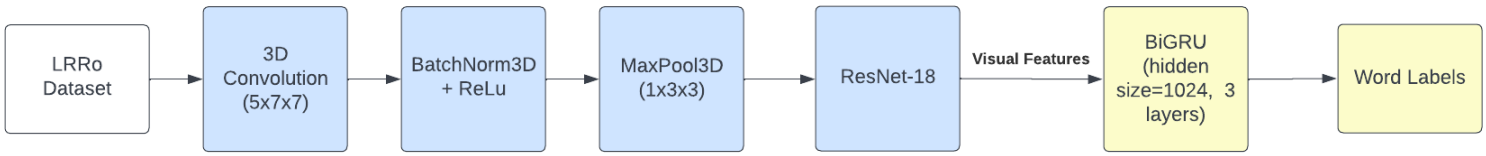


Figure 5: ResNet-18 + GRU model.

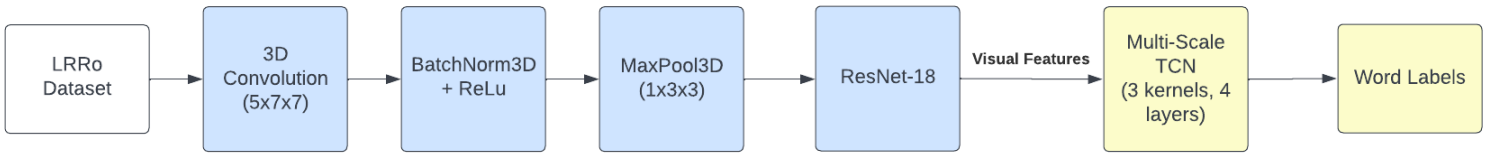


Figure 6: ResNet-18 + MSTCN model.

ResNet-18 + GRU

This model (seen in figure 5) uses a ResNet-18 architecture as the frontend and a bidirectional GRU as the backend. As used in [21], the first convolutional layer of the ResNet has been replaced by a 3D spatio-temporal convolution with a $5 \times 7 \times 7$ kernel and a stride of $1 \times 2 \times 2$, followed by a 3D batch normalization layer, a ReLu layer and finally a max pooling layer which reduces the spatial dimension of the feature maps. The temporal dimension of the input is preserved during processing. Afterwards, the features obtained pass through a global average pooling layer over the height and width dimensions. The GRU network models the sequential relationship between the extracted features.

ResNet-18 + MSTCN

This network (seen in figure 6) uses the same frontend architecture as the previous one and replaces the GRU backend with a multi-stage temporal convolutional network (MS-TCN). The choice of using TCNs is justified by their gradient stability, their low memory consumption during training and their inherent parallelism (as opposed to RNNs which have a sequential nature, computations being performed after their precursors have completed). TCNs have several layers of 1D convolutions which modify the temporal features received as inputs while preserving the sequence length, each layer using a bigger dilation in order to increase the receptive field. The dilation rate is increased exponentially, in order to reduce the number of layers used (and therefore the number of parameters used) using a dilation of 2^i for layer i. MS-TCNs add a layer of complexity on top of that, using more kernels at the same time per layer, in order to combine short and long-term information and obtain more robust feature encodings. More specifically, I use the same MS-TCN setup as [33], with 4 layers and 3 kernels per layer (of size 3, 5 and 7).

Both models are trained for 80 epochs, using Adam as the optimizer. During training, the configuration of parameters with the best accuracy on the validation set is saved, in order to reduce overfitting.

3.2 Optimizations

For both of these models, I employ a series of optimizations which have been proven to improve the performance of lip reading models.

3.2.1 Data Augmentation

As mentioned above, one of the reasons lip reading is a complicated task to solve for deep learning models is the difficulty of obtaining enough training data, which is essential in ensuring that the model learns general features which can be applied in a real-world scenario, on unseen data, and does not just overfit to the training dataset. This is certainly the case for the Wild LRRo dataset, which has only 848 videos in the training set.

Therefore, performance can be improved by modifying the existing data so as to obtain new samples. In my proposed solution, I use two data augmentation techniques in order to increase the number of videos, improve general accuracy and reduce overfitting:

- Horizontal Flip - videos are flipped along the horizontal axis with a probability of 0.5
- Mixup - a new video is created by using a weighted linear combination of two videos from the training set. For a pair of videos and their corresponding word labels (x_i, y_i)

and (x_j, y_j), the mixed-up video is obtained using the following formula:

$$x_{new} = \lambda x_i + (1 - \lambda) x_j$$

$$y_{new} = \lambda y_i + (1 - \lambda) y_j$$

where λ is sampled from the Beta distribution $\lambda \sim \mathcal{B}(0.2, 0.2)$. I chose this distribution for sampling λ since it obtained the best results in the paper written by Zhang et al [34], which introduced Mixup augmentation.

Both are done in place, due to computational restraints. This means that the new samples are generated during training and are replacing the old ones, so as not to occupy too much memory.

3.2.2 Label Smoothing

Generally, classification models are trained to output hard labels (0 for wrong classes and 1 for correct ones). This usually means that the network learns only information about the correct class and becomes overconfident when predicting that class. Instead, it has been proven that classification models perform better when trained to recognize soft probabilities, since the difference between the correct and incorrect labels is not as significant. This acts as a regularization method, helping the network to generalize better.

Label smoothing works by adding a noise distribution which penalizes overconfident predictions:

$$y_{LS} = (1 - \alpha) * y_{true} + \frac{\alpha}{N} \tag{10}$$

where N is the number of classes, and α determines how much the distribution should be smoothed out. In my experiments, I used $\alpha = 0.1$. Figure 7 shows the effect of label smoothing: a noise function is added to the probability distribution of the hard labels, which decreases the weight of the true label and slightly increases the weight of the incorrect ones.

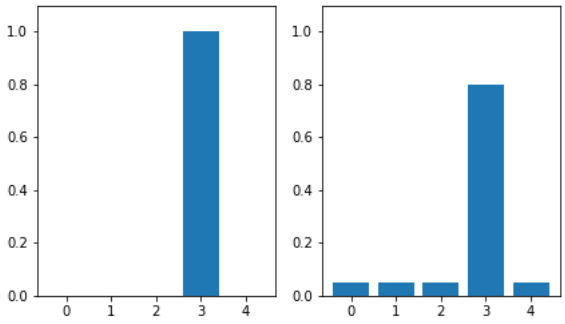


Figure 7: Label Smoothing.

3.2.3 Squeeze-and-Excitation Blocks

Squeeze-and-Excitation networks have been proposed by Hu et al [24]. They improve upon the typical convolutional layer, in which a filter performs operations on all the different input channels and adds the results spatially with equal weights for each channel, in order to produce the output features. Squeeze-and-Excitation (SE) blocks introduce a novel mechanism consisting of a global average pooling and a neural network, in order to assign different weights to each of the input channels ("squeeze"). These values will then be used to scale the filter maps of each convolutional kernel accordingly ("excitation"). They lead to better performance and accuracy while introducing a negligible increase in computational complexity.

In my proposed solutions, I replace the standard residual blocks in the ResNet-18 module with SE blocks (illustrated in figure 8). Note the blue branch, which assigns weights to the different channels. The 2D average pooling reduces each channel to a single scalar value and the two subsequent convolutional layers act as an autoencoder which learns a complex representation of the channel weights. These will be used to scale the features learned by the residual block (the yellow branch) and will be combined with the residual connection (pink branch) by addition. A 1x1 convolution is used as a downsampling layer, for matching the number of channels in the input and output arrays.

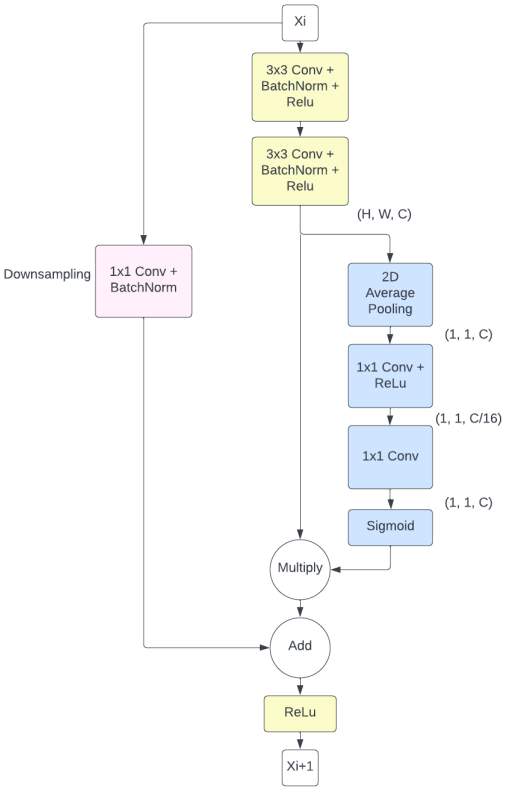


Figure 8: Squeeze-and-Excitation Block used in the models proposed.

3.2.4 Cosine Learning Rate Scheduling

The learning rate is one of the most important hyperparameters to determine for the efficient training of a neural network. For example, if it is too small, the model would need to be trained for a lot of epochs, since each step would barely push the loss towards the local/global minimum. If the learning rate is too big, the model would fail to reach the minimum at all, since the loss would oscillate between nonoptimal values.

To address this, a variable learning rate can be employed. More specifically, for the lip reading models presented above I use a cosine scheduler[1], in which the learning rate rapidly decreases each epoch to a specified minimum value (here *5e-6*), before being increased again and repeating the process. This has the advantages of both low and high values for the learning rate: using high values means that the model does not get stuck in a local minimum, while low values ensure that the model takes small steps towards the optimal minimum it finds. Mathematically, the learning rate is updated using the following formula:

$$\eta = \eta_{min} + \frac{1}{2}(\eta_{init} - \eta_{min})(1 + cos(\frac{t\pi}{T})) \tag{11}$$

where η_{init} is the initial learning rate (here *3e-4*), η_{min} is the lower bound for the learning rate (here *5e-6*), t is the current epoch index and T is the total number of training epochs (usually 80 in my experiments). A plot of the learning rates computed by the cosine scheduler can be seen in figure 9.

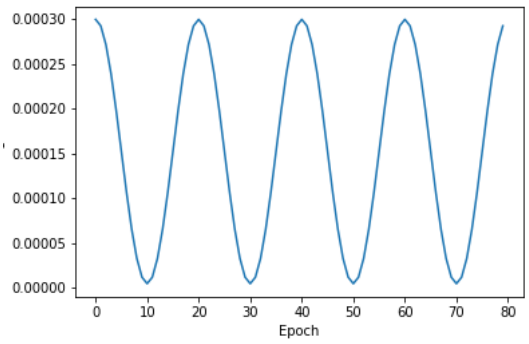


Figure 9: Cosine Learning Rate Scheduler for 80 epochs.

3.2.5 DropBlock Regularization

Dropout is a very effective regularization method for training neural networks that do not overfit to the training data, since it ensures that unique combinations of neurons are used during each iteration, meaning that each neuron learns to generalize better without relying

[1] `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html`

on the neighbouring neurons.

However, dropout has an inherent flaw when used in convolutional neural networks, for which features are correlated spatially. This means that dropping out random activations does not always ensure better generalization, since information about them can be passed on to future layers by the neighbouring activations. Therefore, for convolutional networks, I replaced dropout with a more suitable regularization method for videos and images, namely DropBlock [35], which drops activations in a continuous region in order to ensure that spatial information does not leak into the following layers. Figure 10 shows the difference between Dropout and DropBlock when applied on a 6×6 image.

In line with the observations made in the paper, I combine DropBlock with a linear scheduler in order to reduce the probability of keeping an activation during training. More specifically, I start with a value of 0 for the probability of dropping the activation, which increases during training up to 0.12.

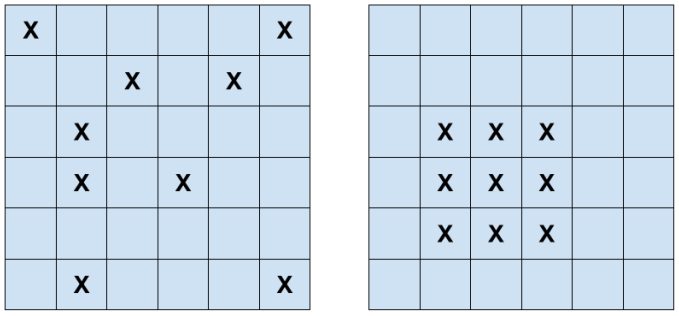


Figure 10: Comparison between Dropout (left) and DropBlock (right).

3.3 Knowledge Distillation

For this section, I used knowledge distillation to transfer knowledge from the ResNet-MSTCN model presented above to a smaller model, which can be easily deployed on devices with less computational power.

Figure 11 presents the standard knowledge distillation model used. Note the replacement of the ResNet-18 frontend network with MobileNetV2 [36], a neural network specifically designed for constrained environments (for example, mobile devices). The multi-stage temporal convolutional network has been replaced by a standard temporal convolutional network with depthwise-separable convolutions, 4 layers and a single kernel of size 5.

I chose MobileNetV2 as the efficient frontend as it was proven to be a robust low-power, low-latency model which uses a small number of parameters while keeping an overall decent performance. The main characteristics of MobileNetV2, which make it suitable for knowledge

distillation, are:

- depthwise-separable convolutions, which split the standard convolution operation into two separate computations: a depthwise convolution which applies a filter for each channel and a pointwise convolution which is basically a 1x1 convolution which creates linear combinations of the channels resulted.
- inverted residuals - which map the low-dimensional inputs of the layer into a feature space with more dimensions, followed by a projection back into the low-dimensional space.
- linear bottlenecks - discarding the nonlinear activation for the projection convolution improves accuracy, since not as much information is lost. A linear shortcut is added between the input and output of the bottleneck.
- ReLu6 - nonlinear activation function which clips the neuron outputs between 0 and 6. Defined as $min(max(0, x), 6)$. It is used to compress the model, since only 3 bits are necessary for the decimal part of the activation.

As for the hyperparameters, several different temperature values will be tested, in order to test the presumption that, due to the considerable difference between the two models, the student is not capable of learning the probability distributions that would result from using higher temperature values. All the models are trained for 200 epochs, to ensure that they have time to learn from both the true labels and the soft targets. During training, the best configuration of parameters is saved based on the validation accuracy (similar to the previous section), in order to avoid overfitting, which is inevitable given the small size of the training dataset.

Because the student is trying to learn the probability distribution of the teacher's soft labels, the Kullback–Leibler divergence is used as a loss function, since it measures the difference between two given probability distributions. Minimizing the Kullback-Leibler divergence will mean ensuring that the soft labels of the student and teacher models are as close as possible. Mathematically, if P and Q are two probaility distributions, the Kullback-Leiber divergence is defined as:

$$KL(Q\|P) = \sum_{x \in X} P(x) \log \frac{Q(x)}{P(x)}. \tag{12}$$

Moreover, as inspired by Furlanello et al. [37], I also employ born-again distillation, in which the student and the teacher models have the same architecture (more specifically, the MobileNetV2-TCN network described above), for two generations, using the best performing model obtained from the standard knowledge distillation process as the first teacher, process which can be seen in figure 12.

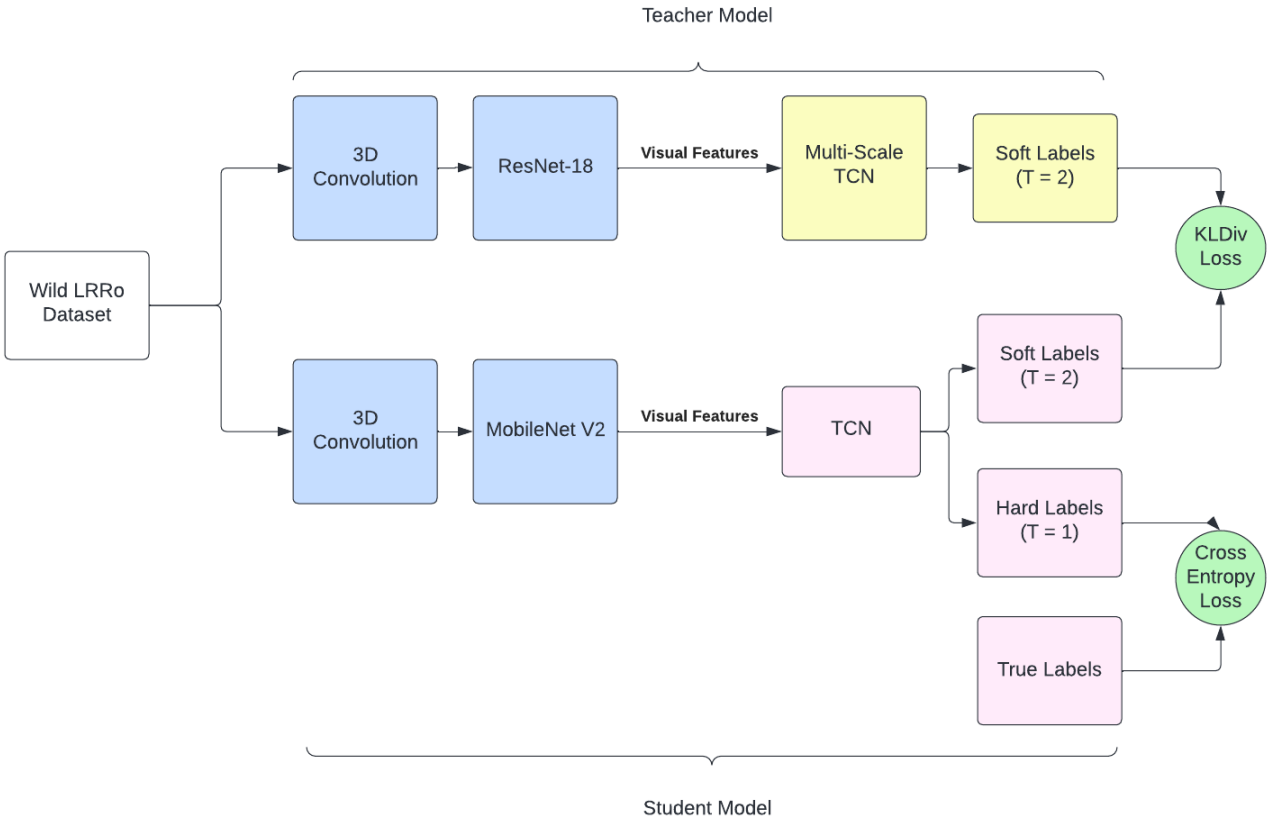


Figure 11: Standard Knowledge Distillation model used.

3.4 Domain Adaptation using GLips

Unsupervised domain adptation, which was proposed by Ganin et al [38], is concerned with improving the performance of a deep learning model on a dataset (named source dataset) by using unlabeled data from a different, but similar one (named target dataset). The model aims to learn a shared representation in the feature space for the two datasets, leveraging uncovered hidden patterns from the target dataset.

The domain adaptation model is trained in an adversarial manner and consists of three different components:

- spatial feature extractor (parameters θ_f) - neural network which will learn language-invariant features for both the source and the target dataset
- label predictor (parameters θ_y) - neural network which will learn to classify sequences of images containing lip movements into word labels
- domain classifier (parameters θ_d) - neural network which will learn to distinguish between the source and the target dataset

Figure 13 presents the architecture for the lip reading model using domain adaptation. Note the two branches the model is split into: one for predicting the word spoken in each video

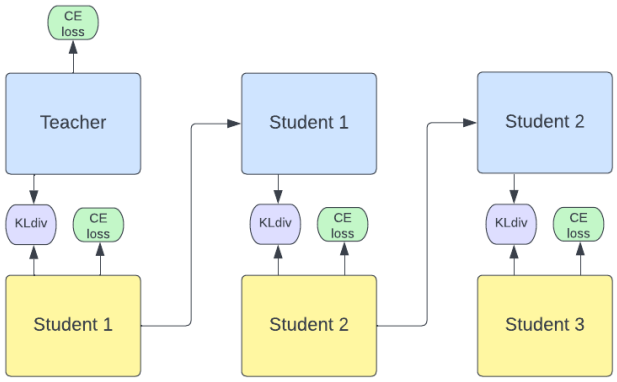


Figure 12: Sequential Knowledge Distillation.

(label predictor) and one for classifying the domain labels (language classifier), both trained simultaneously. Also, a gradient reversal layer was introduced in the language classifier branch, which acts as the identity function during the forward pass, but reverses the sign of the gradients during backpropagation, scaled by a hyperparameter λ:

$$GRL_\lambda(x) = x \tag{13}$$

$$\frac{\partial GRL_\lambda}{\partial x} = -\lambda * \mathbf{I} \tag{14}$$

where $\mathbf{I}$ is the identity matrix.

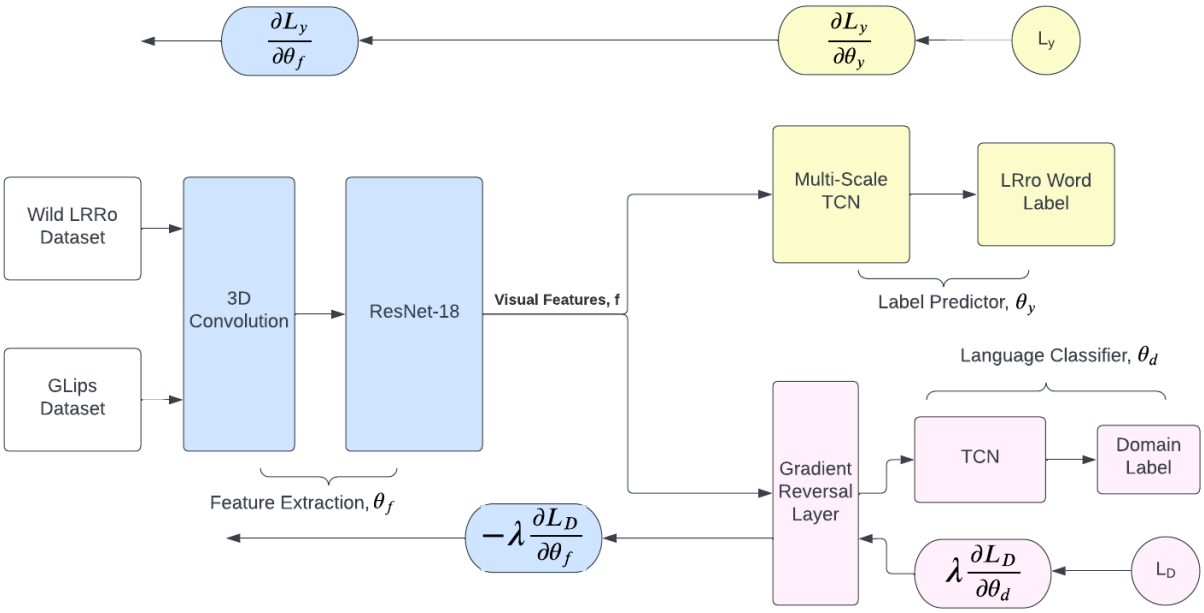


Figure 13: Domain Adaptation Lip Reading Model used.

The entire model's loss is therefore defined as follows:

$$L_{DA}(\theta_f, \theta_y, \theta_d) = L_y(\theta_f, \theta_y) - \lambda L_d(\theta_f, \theta_d) \tag{15}$$

where L_y is the lip reading predictor's loss, L_d is the domain classifier's loss, which is weighted by the parameter λ. Because of the gradient reversal layer, minimizing the loss L_{DA} implies maximizing L_d, forcing the feature extractor into learning representations that the domain classifier can't differentiate. L_d will be calculated for both datasets, while L_y will be computed only for the Romanian dataset. Thus, the model will use information learned from an unlabeled domain (that is, not containing word labels) to improve its performance on a related, but different dataset. Mathematically, the parameters of the model will be updated during gradient descent as follows:

$$\theta_f = \theta_f - \alpha * (\frac{\partial L_y}{\partial \theta_f} - \lambda * \frac{\partial Ld}{\partial \theta_f}) \tag{16}$$

$$\theta_y = \theta_y - \alpha * \frac{\partial L_y}{\partial \theta_y} \tag{17}$$

$$\theta_d = \theta_d - \alpha * \frac{\partial L_d}{\partial \theta_d} \tag{18}$$

For the lip reading task, Wild LRRo is used as the source dataset and GLips as the target dataset. Because the size of the original GLips dataset is significantly bigger than LRRo (approximately 250 000 entries, compared to just 1000 in the Romanian dataset), I normalized the former to have 848 samples in the training set and 120 in the validation set. The test set is not needed since the performance of the model will be evaluated only on the LRRo dataset. The reasoning behind using data from another language is that the lip shapes and visemes are similar for different languages, allowing the model to learn more robust visual representations. This is very important for small datasets, and particularly LRRo, for which the risk of overfitting is bigger.

4 RESULTS AND EXPERIMENTS

4.1 Implementation Details and Environment

The solutions presented above have been implemented and tested using Google Colab [1], which is based on Jupyter notebooks and offers access to high-performance storage and computing resources, in the cloud. Most notably, they provide GPUs, which drastically improve the training time of deep learning architectures, due to their ability to perform multiple computations in parallel (they have thousands of computing units, as opposed to regular CPUs which usually have at most 16). Hardware specifications of the Colab VMs used:

- Intel Xeon CPU, 2.30GHz, Dual-Core
- 13GB RAM
- 167GB storage memory
- NVIDIA T4 GPU - 16GB memory, 8.1 TFLOPS single precision performance, 2560 CUDA cores
- NVIDIA P100 GPU - 16GB memory, 9.3 TFLOPS single precision performance, 3584 CUDA cores

All the models presented above have been tested on the Wild LRRo dataset, for two reasons:

- performance on it is much more representative of real world usage scenarios, because videos have variable angles, lighting conditions, emotional and physical profiles of speakers
- computational resources are scarce, Google Colab not allowing background model training and even limiting GPU usage. The Lab LRRo dataset is almost 8 times larger, so training on it would take several hours.

4.2 Metrics

Metrics used to assess the performance of the models are:

- accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

[1] `https://colab.research.google.com`

- precision:

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

- recall:

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

- F1-score:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{22}$$

- top-5 accuracy: the correct label can be found among the top 5 predictions made by the model. It is a particularly useful metric for lip reading, given the many-to-one mapping between visemes and phonemes.

where:

- TP (true positives) = the model correctly classified an entry as the positive class
- TN (true negatives) = the model correctly classified an entry as the negative class
- FP (false positives) = the model incorrectly classified a negative entry as the positive class
- FN (false negatives) = the model incorrectly classified a positive entry as the negative class

These metrics have been proposed for binay classification (classifying a sample as being either positive or negative), but they can be generalized for multi-class classification by considering the correct class as the positive one and all other classes as the negative one. Generally, there is a trade-off between precision and recall (when precision increases, recall decreases and vice versa). Therefore, F1-score is a more important metric to observe, since it provides a means of analyzing both precision and recall simultaneously using a single numerical value.

Moreover, loss and accuracy plots for the training and validation sets will be provided, in order to determine if the results obtained are relevant or not (for example, the models may be overfitting to the training data, failing to generalize on unseen entries). For the most efficient models, confusion matrices will be generated and an analysis of the most difficult classes to classify will be made.

4.3 Dataset Analysis

In order to have a clear perspective on the results, it is important to understand the distribution of the labels in the dataset. Figure 14 shows that some labels are underrepresented in all three

subsets, for example "lucrurile" ("things"), "cincizeci" ("fifty") and "romana" ("Romanian"), so it is expected that the models trained on them will have trouble classifying them, since they will most likely overfit. The most frequent words in the dataset are "aceasta" (Romanian word for "this"), "dumneavoastra" (polite pronoun replacing the informal pronouns "tu" and "voi", meaning "you"), "scoala" (meaning "school") and "Romania", which is explained by the fact that they are common words heard on Romanian television shows.

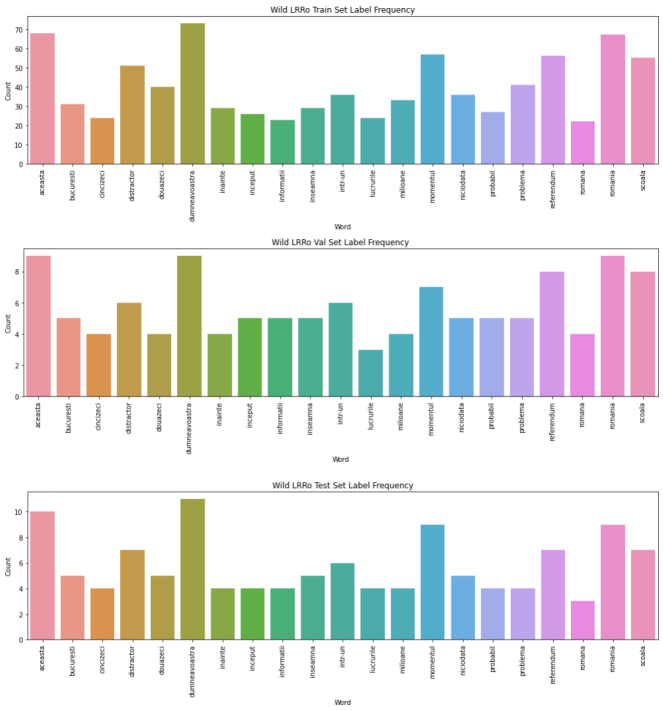


Figure 14: Label Frequencies for each subset of Wild LRRo.

4.4 Standard Lip Reading Models

Given the small size of the Wild LRRo dataset, I expected regularization to be an important factor in the performance of the lip reading architectures, since it is easier for the model to learn to recognize samples, rather than learn underlying patterns that generalize on unseen data. Since all other regularization methods have been proven to perform well on the lip reading models, I place particular emphasis on the impact of DropBlock regularization, since, to the best of my knowledge, it has not been yet tested on this task.

Results on the test set can be seen in Table 1. All the values shown are percentages (%). One can see that adding DropBlock regularization improves performance by 5.8% for Resnet-MSTCN and 3.3% for the Resnet-GRU network. It is also important to notice the impact of the backend module on all metrics, the multi-stage temporal convolutional network surpassing the GRU frontend by almost 1% on top-1 accuracy and 4.13% on top-5 accuracy. Table 2

Frontend	Backend	DropBlock	Accuracy	Top-5 Acc	Precision	Recall	F1-Score
ResNet-18	**MSTCN**	**Yes**	**42.14**	**71.07**	**40.15**	40.27	**38.8**
ResNet-18	**MSTCN**	**No**	36.36	64.46	32.98	34.85	32.62
ResNet-18	**GRU**	**Yes**	41.32	66.94	38.68	**43.08**	38.46
ResNet-18	**GRU**	**No**	38.01	62.80	36.40	39.63	36.17

Table 1: Results for the standard lip reading models, with and without DropBlock regularization.

also shows that the GRU frontend requires 7 times more memory for running (the forward size) and is only two times smaller in size, even though it has 35 times less parameters.

Model	Forward/Backward Size (MB)	Total Size (MB)	Parameters
ResNet + MSTCN	145.06	1593.5	379.4M
ResNet-18 + GRU	829.56	873.59	11.3M

Table 2: Storage and memory comparison between Resnet + MSTCN and Resnet + GRU models.

Figure 15 provides further visualization of the difference between the two best performing models, highlighting that they perform similarly but that the model with the GRU frontend converges faster and is overall more stable in its performance.

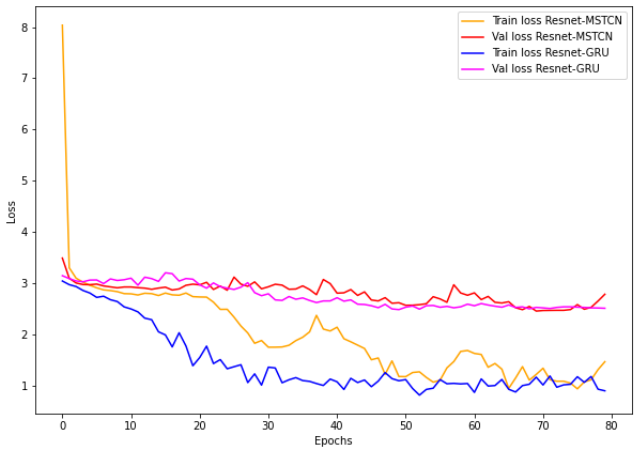


Figure 15: Train and Validation losses for the Resnet-GRU and Resnet-MSTCN models.

The confusion matrix for the Resnet-MSTCN model, with DropBlock regularization, is shown in figure 16. As can be seen, the model has problems classifying the following words: "aceasta", "bucuresti", "problema", "romana" and "distractor", while better recognizing the words: "dumneavoastra", "momentul", "Romania" and "scoala", which is in accordance with the observations made in the section Dataset Analysis. Also note that there are certain pairs

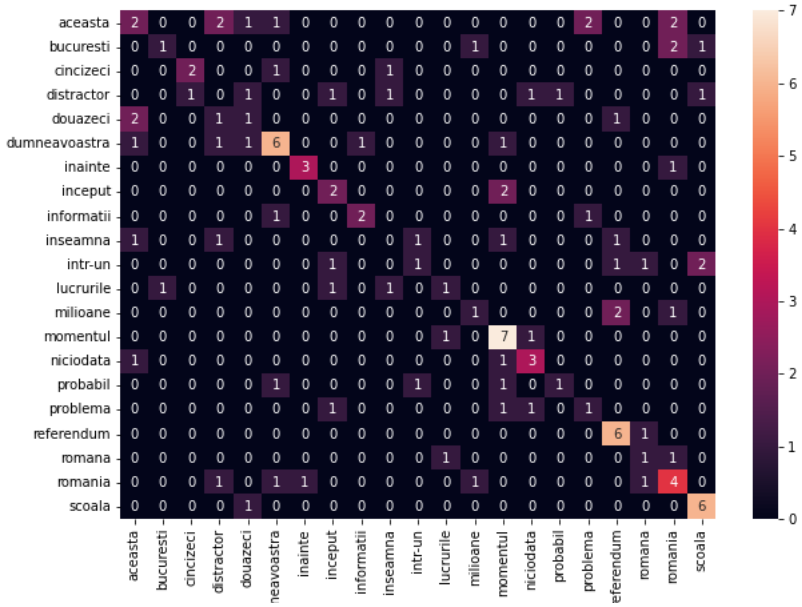


Figure 16: Confusion Matrix for the ResNet-MSTCN model, with DropBlock regularization.

of words that the model mistakes: "inceput" and "momentul", "bucuresti" and "romana", "intr-un" and "scoala".

4.5 Knowledge Distillation

4.5.1 Standard Distillation

Table 3 shows a comparison between the teacher and the student, where MAC (multiply-accumulate operation) represents the operation of multiplying two numbers and adding the result to an accumulator. Roughly, the number of FLOPS equals twice the number of MAC operations. One can notice that the teacher executes 97% more operations , has approximately 96.75% more parameters and is 15.5 times bigger in size than the student. The computational information of the models was obtained using the *ptflops* library[2].

Model	Number of Parameters	Size (MB)	MAC
ResNet-18 + MSTCN	379.5M	1593.50	26.13G
MobileNet-V2 + DSTCN	12.3M	102.36	801.35M

Table 3: Teacher-student comparison

I tested different hyperparameter configurations for this experiment, in order to decide which one would lead to the best performance (student metrics as close to the teacher as possible).

[2] `https://github.com/sovrasov/flops-counter.pytorch`

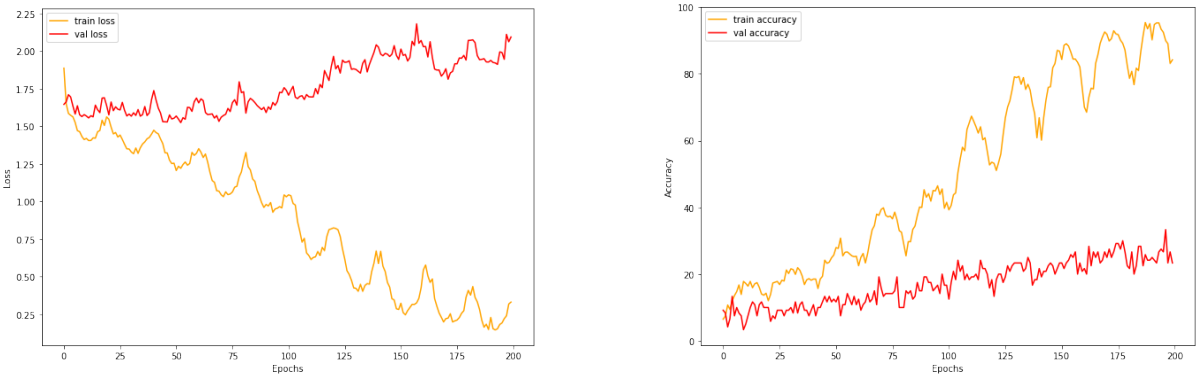
Results can be seen in table 4. Two experiments were carried out in which the hyperparameter values changed during training:

- $\alpha = 0.5, T = 2$ for the first 80 epochs, followed by $\alpha = 0.9, T = 5$ for the next 120 epochs (Mixed-1 in the table)
- $\alpha = 0.9, T = 2$ for the first 80 epochs, followed by $\alpha = 0.5, T = 5$ for the next 120 epochs (Mixed-2 in the table).

Configuration	Accuracy	Top-5 Acc	Precision	Recall	F1-Score
$\alpha = 0.7, T = 2$	33.05	58.67	31.42	31.28	30.67
$\alpha = 0.3, T = 3$	31.40	**61.98**	29.08	27.75	27.75
$\alpha = 0.5, T = 2$	30.57	61.15	28.37	27.45	27.06
$\alpha = 0.5, T = 5$	32.23	55.37	30.29	31.14	28.62
$\alpha = 0.5, T = 10$	**36.36**	60.33	**32.72**	**38.28**	**32.56**
Mixed-1	31.40	51.23	26.32	28.37	25.80
Mixed-2	28.09	59.50	23.27	21.93	22.31

Table 4: Results for the different hyperparameters configurations used for distilling knowledge

Note that the best hyperparameter configuration is $\alpha = 0.5, T = 10$, contrary to the prior expectation that a lower temperature value would lead to a better student performance, due to the substantial size difference between the two models. Its loss and accuracy plots can be seen in figure 17. They show an obvious downside of this technique, which is that the model rapidly entered a state in which it was overfitting to the training set: the validation loss started increasing at around the 100 epoch mark while the training loss kept going down. This is also evident in the accuracy plot on the right, where even though the validation accuracy seems to be on an ascending path, it goes up at a much slower pace than the training accuracy. I believe this is caused by the considerable difference between the two models, the student simply not being capable of learning the same complex features as the teacher.



(a) Loss

(b) Accuracy

Figure 17: Loss and accuracy plots for the best performing Knowledge Distillation model.

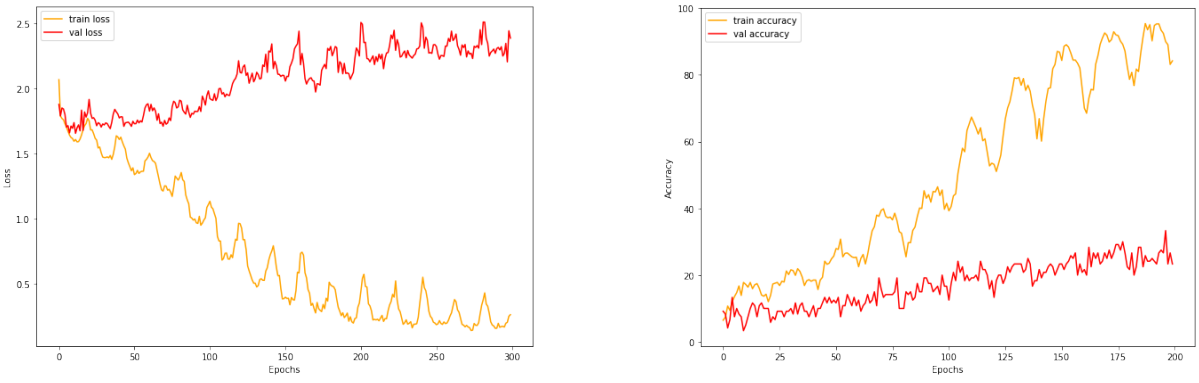
4.5.2 Born-Again Distillation

The best model obtained during the standard knowledge distillation process was used as the first teacher model for the born-again distillation experiment, which means that both the student and teacher models have the same architecture. The hyperparameters used were $\alpha = 0.5$ and $T = 10$, since they lead to the best performance in the previous experiment. Results of born-again distillation can be seen in table 5. The student model in each generation was trained for 300 epochs, saving only the model with the best validation accuracy. As can

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score
Teacher	36.36	**60.33**	**32.72**	**38.28**	**32.56**
Student - Gen 1	**38.01**	58.67	28.37	27.45	27.06
Student - Gen 2	36.06	57.45	27.45	27.13	27.29

Table 5: Results for Born-Again Distillation

be seen, only one generation of training brought an improvement in top-1 accuracy. However, the student models are unstable, as shown by the fact that the overall performance on all other metrics seems to be getting worse. I suspect this is because of the hyperparameters used, as well as the fact that the teacher already overfitted to the training set (as explained above). The loss and accuracy plots for the model obtained after one generation of knowledge distillation can be seen in figure 18, showing a similar trend of overfitting as the model trained above.



(a) Loss

(b) Accuracy

Figure 18: Loss and accuracy plots of the model after born-again distillation.

4.6 Domain Adaptation using GLips

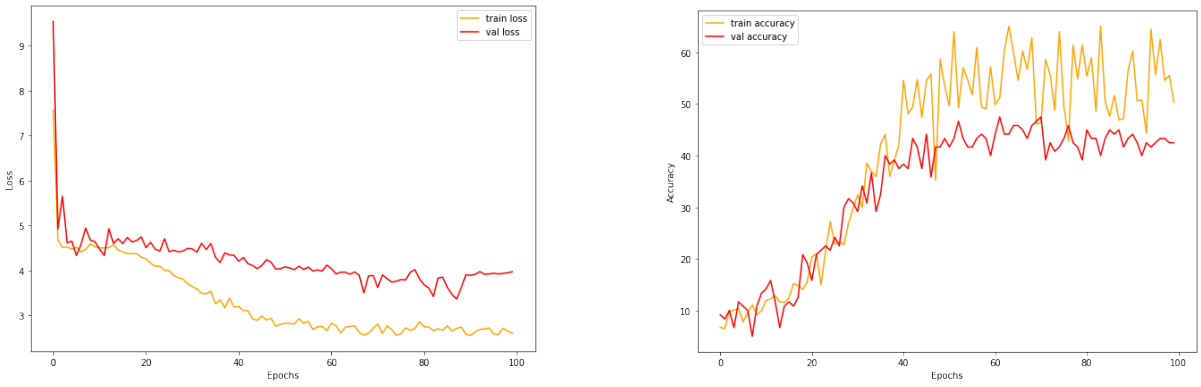
Table 6 shows a comparison between the models with the best performances from the previous two sections, to which I added the results of the model trained adversarially using the GLips

dataset. The results show that using domain adaptation improves results significantly, even if no labels from the target dataset are used: top-1 accuracy increased by 8%, recall by 9% and precision by almost 6%, only top-5 accuracy showing a slight 1.7% increase, which is still substantial, given the difficult nature of the task.

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score
ResNet + MSTCN	42.14	71.07	40.15	40.27	38.8
MobileNetV2 + DSTCN	38.01	58.67	28.37	27.45	27.06
ResNet + MSTCN + GLips	**50.41**	**72.72**	**49.39**	**46.01**	**45.04**

Table 6: Metrics for the best performing models on all three categories (Standard Deep Learning, Knowledge Distillation, Domain Adaptation

The loss and accuracy plots during training can be seen in figure 19. The close train and validation losses show that the model learns a good representation which generalizes well on unseen data, particularly due to the several regularization methods used. As can be seen, there is a slight tendency for the validation loss to decrease, which implies that training for longer epochs could lead to even better performance.



(a) Loss

(b) Accuracy

Figure 19: Loss and accuracy plots for the model trained using Domain Adaptation.

The confusion matrix for the Resnet-MSTCN model, trained using domain adaptation from the GLips dataset, is shown in figure 20. It can clearly be seen that this model outperforms the standard lip reading architectures, since it makes less incorrect predictions. For example, for the word "dumneavoastra", the model makes 10 accurate predictions, as opposed to the ResNet-MSTCN model described above, which was accurate only 6 times. It also learned to recognize the word "Romania" better, doubling its number of correct guesses. It, however, still has trouble recognizing the words "distractor", "romana", "problema" and "inseamna", just like its counterpart. Note also that some of the problematic pairs from the standard model appear here as well: "Bucuresti" and "scoala", or "aceasta" and "dumneavoastra".

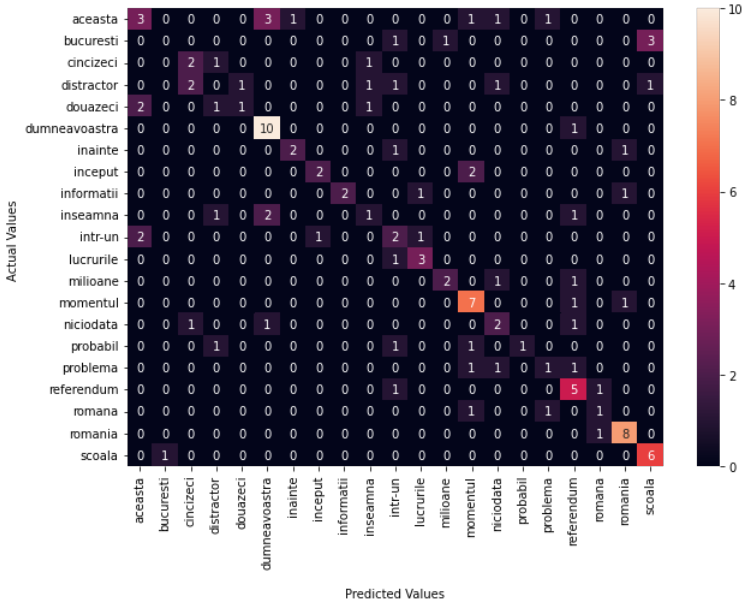


Figure 20: Confusion Matrix for the ResNet-MSTCN model, with Domain Adaptation.

Moreover, figure 21 shows a visualization of the features learned by the domain adaptation model. It clearly demonstrates that adding the language classifier branch and training it adversarially help the model to learn indistinguishable features for both the Romanian and the German dataset. These were obtained using t-distributed stochastic neighbor embedding (t-SNE) [3], which is an effective dimensionality reduction technique used to project a high-dimensional dataset onto a 2 or 3-dimensional plane.

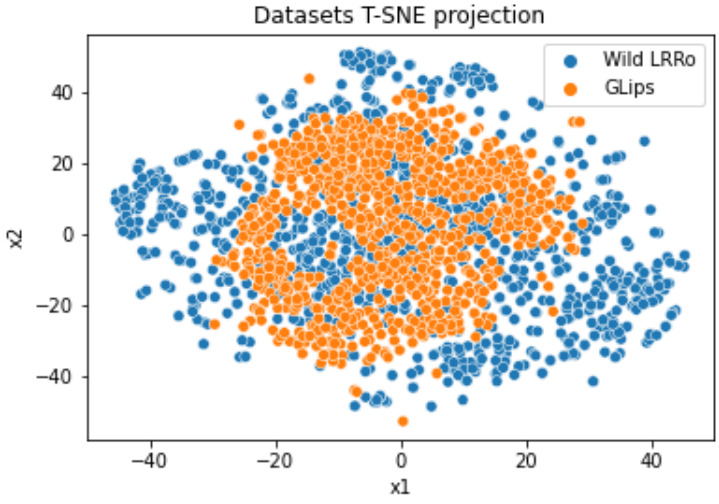


Figure 21: Visualization of the features learned by the Domain Adaptation Model.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

4.7 Final Results and Comparison

Table 7 shows the top-1 and top-5 accuracy of the models trained and tested above, compared with the baseline performance obtained by Jitaru et al. [31] in their paper proposing the LRRo dataset. More specifically, they used two convolutional neural network models: a VGG-M model [39] with multiple towers, similarly to the paper introducing the LRW dataset [16], and an Inception-V4 model [40].

Model	Top-1 Accuracy	Top-5 Accuracy
VGG-M + MT	33.00	61.00
Inception-V4	33.00	62.00
ResNet + MSTCN	42.14	71.07
ResNet + GRU	41.32	66.94
MobileNetV2 + DSTCN	36.36	60.33
KD Student - Gen 1	38.01	58.67
ResNet + MSTCN + GLips	**50.41**	**72.72**

Table 7: Accuracy comparison between the models proposed in this paper and the state-of-the-art architectures. All results are reported on the Wild LRRo datset.

Results clearly show that all the architectures implemented in this paper outperform the baseline models in terms of top-1 accuracy. Models trained using knowledge distillation obtained better top-1 accuracies and similar top-5 accuracies. The best performance is obtained by the Domain Adaptation model, which surpasses the state-of-the art results by 17% and 11%, respectively, which is a significant outcome since it shows how important transfer learning can be in learning robust features for smaller datasets, prone to overfitting.

5 CONCLUSIONS AND FURTHER WORK

In this paper, I implemented three advanced deep learning techniques for the task of lip reading, which means that neural networks are being trained to recognize the word(s) being spoken in a video, solely using the visual information of the frames and of the speaker's lip movements. All three of these methods have been tested on the recently proposed and underrepresented Wild LRRo dataset, which contains videos taken from Romanian TV shows and news programs, meaning that they were taken in an uncontrolled environment, with variable camera angles, lighting conditions and speech characteristics. Therefore, it poses a real challenge to neural network models, especially considering the difficult nature of the task (human baseline performance has been proven to be relatively low).

The three methods proposed are:

- standard lip reading neural networks with a ResNet-18 as the frontend module and two different types of backends (GRUs and multi-stage temporal convolutional networks). Considering the small size of the dataset, I placed particular emphasis on regularization methods and general optimizations, such as data augmentation, label smoothing, squeeze-and-excitation residual blocks and DropBlock instead of dropout in convolutional layers.
- knowledge distillation, in which a smaller model consisting of a MobileNetV2 frontend and a small temporal convolutional network backend was trained using the predictions of the best performing standard model as targets, normalized using a temperature value to ensure that the network learns class correlations from the teacher. Several hyperparameter configurations have been tested, in order to decide the most efficient combination. I also carried out experiments on born-again distillation, in which both the student and the teacher have the same architectures.
- domain adaptation, in which I trained a model adversarially in order to ensure that it learns language-invariant features, using the recently proposed German language GLips dataset.

As far as results are concerned, I showed that MS-TCN backends outperform GRUs in both accuracy and resources consumed, although GRUs seemed to converge faster and be more stable during training. Regularization methods, and particularly DropBlock, have been proven to be highly effective, improving general accuracy by as much as 6% for the temporal convolutional model. The knowledge distillation experiments showed that smaller models can be trained to perform decently while using 97% less resources, paving the way towards deploying efficient lip reading models on mobile devices. Born-again distillation results were unstable and generally

unreliable, due to the models' tendency to overfit early during training. The most important result, however, was proving that using an unlabeled dataset from the relatively different and unrelated German language can drastically improve the performance of the models on the Romanian dataset, result which is important since it questions the common consensus that large volumes of data are absolutely necessary for robust, high-end, state-of-the-art models.

Further work includes improving the generalization of the models by adding more regularization techniques, especially for the small models trained with knowledge distillation, which have overfitted the most due to their small number of parameters and the limited size of the dataset used. This could include more data augmentation methods (rotation, scaling, brightness variation, adding noise to the frames), Stochastic Depth combined with training for more epochs, or testing different optimizers (AdamW, LAMB, SGD with momentum). Sequential knowledge distillation, in which smaller and smaller models are trained in generations, could also be used. Moreover, for the domain adaptation experiment, using more discriminators based on different characteristics of the videos used (language, speaker age and gender) could lead to more robust features. Lastly, the models can be deployed in a mobile or web application, which would have to separate and classify the different words being spoken in real-time.

BIBLIOGRAPHY

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[4] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Convolutional neural networks. `https://www.ibm.com/cloud/learn/convolutional-neural-networks`. Accessed: 2022-06-03.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[8] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[12] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[13] Sarah Hilder, Richard W Harvey, and Barry-John Theobald. Comparison of human and machine-based lip-reading. In *AVSP*, pages 86–89, 2009.

[14] Antti J Eronen, Vesa T Peltonen, Juha T Tuomi, Anssi P Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):321–329, 2005.

[15] Stéphane Dupont and Juergen Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE transactions on multimedia*, 2(3):141–151, 2000.

[16] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In *Asian conference on computer vision*, pages 87–103. Springer, 2016.

[17] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *Asian conference on computer vision*, pages 251–263. Springer, 2016.

[18] Iryna Anina, Ziheng Zhou, Guoying Zhao, and Matti Pietikäinen. Ouluvs2: A multi-view audiovisual database for non-rigid mouth motion analysis. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–5. IEEE, 2015.

[19] Yannis M Assael, Brendan Shillingford, Shimon Whiteson, and Nando De Freitas. Lipnet: End-to-end sentence-level lipreading. *arXiv preprint arXiv:1611.01599*, 2016.

[20] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America*, 120(5):2421–2424, 2006.

[21] Themos Stafylakis and Georgios Tzimiropoulos. Combining residual networks with lstms for lipreading. *arXiv preprint arXiv:1703.04105*, 2017.

[22] Shuang Yang, Yuanhang Zhang, Dalu Feng, Mingmin Yang, Chenhao Wang, Jingyun Xiao, Keyu Long, Shiguang Shan, and Xilin Chen. Lrw-1000: A naturally-distributed large-scale benchmark for lip reading in the wild. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–8. IEEE, 2019.

[23] D Feng, S Yang, S Shan, and X Chen. Learn an effective lip reading model without pains. arxiv 2020. *arXiv preprint arXiv:2011.07557*.

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[25] Nilay Shrivastava, Astitwa Saxena, Yaman Kumar, Rajiv Ratn Shah, Debanjan Mahata, and Amanda Stent. Mobivsr: A visual speech recognition solution for mobile devices. *arXiv preprint arXiv:1905.03968*, 2019.

[26] Ya Zhao, Rui Xu, Xinchao Wang, Peng Hou, Haihong Tang, and Mingli Song. Hearing lips: Improving lip reading by distilling speech recognizers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6917–6924, 2020.

[27] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Asr is all you need: Cross-modal distillation for lip reading. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2143–2147. IEEE, 2020.

[28] Pingchuan Ma, Brais Martinez, Stavros Petridis, and Maja Pantic. Towards practical lipreading with distilled and efficient models. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7608–7612. IEEE, 2021.

[29] Andrei-Cosmin Jitaru, Liviu-Daniel Ştefan, and Bogdan Ionescu. Toward language-independent lip reading: A transfer learning approach. In *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, pages 1–4. IEEE, 2021.

[30] Mohammed Abdelwahab and Carlos Busso. Domain adversarial for acoustic emotion recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2423–2435, 2018.

[31] Andrei Cosmin Jitaru, Şeila Abdulamit, and Bogdan Ionescu. Lrro: a lip reading data set for the under-resourced romanian language. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 267–272, 2020.

[32] Gerald Schwiebert, Cornelius Weber, Leyuan Qu, Henrique Siqueira, and Stefan Wermter. A multimodal german dataset for automatic lip reading systems and transfer learning. *arXiv preprint arXiv:2202.13403*, 2022.

[33] Brais Martinez, Pingchuan Ma, Stavros Petridis, and Maja Pantic. Lipreading using temporal convolutional networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6319–6323. IEEE, 2020.

[34] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[35] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018.

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[37] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018.

[38] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[39] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[40] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.