Lip Reading and Natural Language Processing

Aleem Alibhai*
Heidar Kourosh Davoudi*
aleem.alibhai@ontariotechu.net
heidar.davoudi@ontariotechu.net
Ontario Tech University
Oshawa, Ontario, Canada

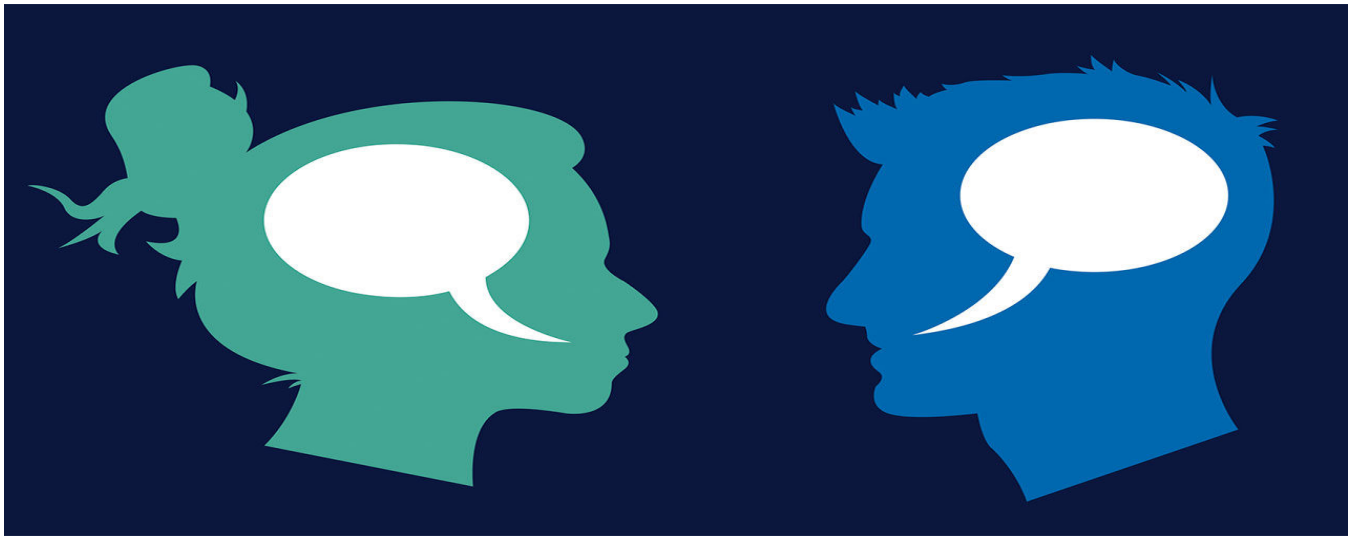

Figure 1. Say it with your lips

Abstract

Lipreading is a subset of Automatic Speech Recognition that focuses on determining speech based solely on the movements of the speaker's mouth. Recent attempts to automate lipreading make use of Spatiotemporal Convolutional Neural Networks and Recurrent Neural Networks to first learn letter symbols and then make predictions. These methods afford the model temporal information, allowing it to use the context of the word when predicting letters. Context in lipreading however is not limited to the specific word being uttered. A large part of human lipreading is the relationship between the word being spoken and the other words in the sentence. It is much easier to predict the next word in a sequence when the lip reader has knowledge of previous words uttered. Our proposed model attempts to afford automated systems with the same grammatical and semantic context that human lip readers are afforded using a word-level language model in the decoding step.

Keywords: spatiotemporal convolutional neural networks, recurrent neural networks, natural language processing, word beam search, connectionist temporal classification loss

ACM Reference Format:
Aleem Alibhai and Heidar Kourosh Davoudi. 2021. Lip Reading and Natural Language Processing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Lipreading is used by many groups of people for many reasons. Primarily, lipreading is the main source of communication for the hearing impaired. Lipreading can also be used in environments where there is a large amount of ambient noise, such as public speeches or events. Here lipreading

*This work was done as an undergraduate thesis by Aleem, supervised by Dr. Davoudi

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

can be used to supplement audio to increase speaker understandability. Automatic lipreading can allow all people to take advantage of this increased clarity without having to personally learn to lipread.

Lipreading is a difficult task for humans let alone machines. Many mouth movements and positions (visemes) map to multiple different sounds (phonemes). Without context, these visemes are very difficult to disambiguate. Previous works such as LipNet [1] attempt to address this by providing context to the automatic lipreading system by utilizing recurrent units to produce sentence-level predictions. This allows the network to utilize the history of character predictions for subsequent predictions in the sentence. Although this model has sentence output, internally it operates on a character level. This leaves the network lacking some of the semantic and grammatical contexts that are afforded in word-level architectures.

The GRID dataset [3], used for LipNet uses a simple sentence structure for all of its samples in the form command, color, preposition, letter, digit, adverb. This simple grammar removes the need for the system to understand grammar. Though the model may perform well in these cases, the GRID dataset is not representative of real-world speech. We suspect that when using real-world data, this model will begin to show its lack of semantic and grammatical awareness. Homonyms, where visemes and phonemes are identical, are likely to be a large source of error without grammatical context.

In this paper, We propose the addition of Word Beam Search (WBS) to the LipNet architecture, affording it the grammatical and semantic contexts it needs to accurately lipread real-world data samples. WBS incorporates a word-level language model when decoding the network's output into sentences. This allows the model to maintain all of the benefits of the character level model while also affording it grammatic insight in the form of Ngrams and forecasting.

We test our model with four beam evaluation methods and compare the results to LipNet's performance.

2 LipNet

Character and word-level models have often been used to tackle lipreading. These models however have no reference to the surrounding text that their input occurs in. They, therefore, are unable to utilize any grammatical or semantic context in their predictions. Sentence level models do not face this same limitation; their input must follow the specific grammatical and semantic rules of their dataset. This gives us an opportunity to have the model learn these rules and apply them to their predictions. For our work, we use the sentence-level LipNet [1] model as our starting point. We use the PyTorch implementation created by the original authors.

2.1 Architecture

LipNet takes a preprocessed video clip as input and attempts to map the video to a text sequence. Preprocessing includes:

- Splitting the video into a chronological sequence of frames.
- Detecting and cropping the mouth region.
- Resizing the cropped image to a standard size.
- Performing flips on the frames (in training).

2.1.1 3D Convolution Layers. 3D convolutional neural networks, desribed by [6] perform the same task as their two-dimensional counterparts. 2D convolution layers perform feature extraction by taking the weighted average of pixels inside a kernel of shape (k_{width}, k_{height}). These convolution layers can also be used to decrease the resolution of the image by using a stride greater than 1. 3D convolution layers perform the same feature extraction, but convolve along the temporal axis as well. Their kernels are of shape (k_t, k_{width}, k_{height}), where kt is the number of frames or timesteps being convolved across.

Inputs are first passed through the spatiotemporal convolutional neural network (STCNN). The STCNN is made up of three 3D convolution layers, each preceding a spatial pooling layer to perform feature extraction.

2.1.2 Bidirectional Gated Recurrent Units. Gated Recurrent Units (GRU) are a form of recurrent neural networks similar to Long Short-Term Memory units dscribed in [2]. They are used to denote which timesteps contain important information to pass through to their output and which inputs should be forgotten. Bidirectional GRUs (BiGRU) perform this operation both in the forward direction and in the reverse direction. This ensures that each timestep is considered with the context of both previous and following elements.

The features extracted from the STCNN are processed by two BiGRU layers. This leaves us with a set of relevant features for each timestep. We pass these features to a linear transformation layer with output dimensionality of 28 (26 letters + space + CTC blank). This leaves us with the final character probabilities for each timestep.

2.2 Connectionist Temporal Classification Loss

To combat the need for perfectly aligned input and output, LipNet trains the model with Connectionist Temporal Classification (CTC) loss from [4]. CTC computes the probability of the target string by summing all sequences that are output by the model that are equivalent to it. This equivalence is defined by first removing all instances of duplicate letters in the output sequence, then removing all instances of the CTC blank. If the resulting sequence is identical to the target, the two are deemed equivalent. The CTC blank is added to separate instances where double letters in the word exist (e.g. letter). If a CTC blank appears between two occurrences of the same letter, we know a double letter is present. Because

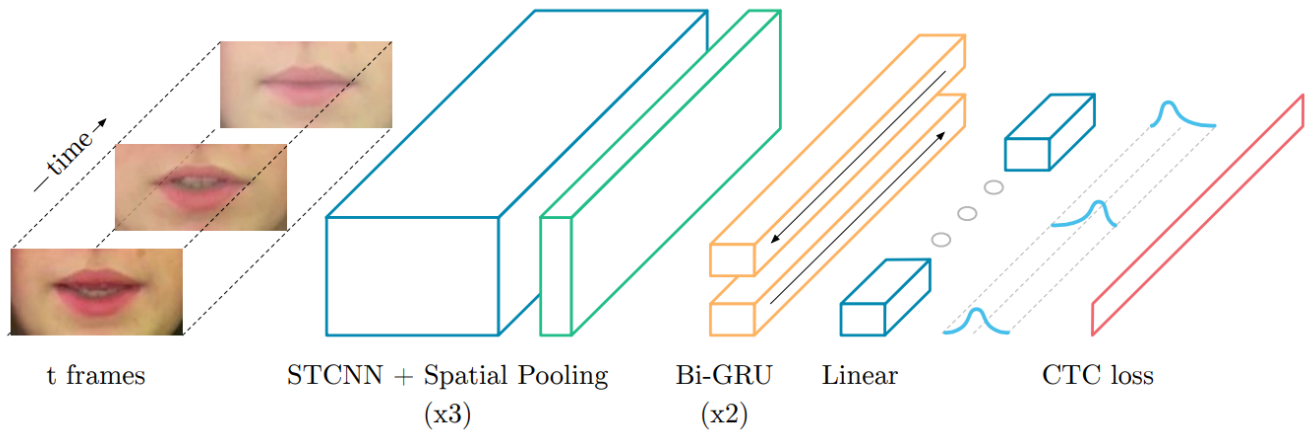

Figure 2. LipNet architecture. A sequence of T frames is used as input, and is processed by 3 layers of STCNN, each followed by a spatial max-pooling layer. The features extracted are processed by 2 Bi-GRUs; each time-step of the GRU output is processed by a linear layer and a softmax. This end-to-end model is trained with CTC. From [1]

a longer output sequence can be reduced to a shorter string, we can use variable lengths of input frames. Using CTC loss, therefore, eliminates the need for perfectly aligning input frames to specific letters at each timestep as well as matching sequence lengths. Instead, the output of the model can be directly compared to the target string.

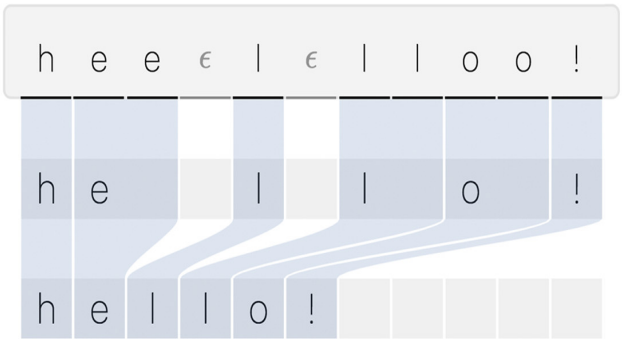

Figure 3. Visual example of the CTC loss function collapsing input text from [5]

2.3 Challenges with LipNet

As a whole LipNet is a sentence-level lip reading model however, internally it works on a character level. Each frame is mapped to its corresponding character and these characters are combined to make the output sentence. The BiGRU layers in LipNet's model learn a pseudo character language model, understanding that the prediction of a character is highly dependent on the context of its previous and following characters. This allows the model to correctly predict the spelling of words and even catch silent letters. This however does not extend to the words being said. The model can successfully separate the input sequence into distinct words but knows nothing of their meaning. The model has no grammatical or semantic context for the words it is predicting. Without this context, the model is left without the tools to disambiguate words that are made up of similar visemes or phonemes. Homonyms such as their, there, and they're, all share the same visemes and phonemes. Disambiguating between them would be very difficult for LipNet as their difference is defined not by character level interaction, but by the semantic context in which they appear.

LipNet performs well on the GRID dataset as it is made up of a very simple sentence structure. In this case, the model still struggles with characters or words with similar visemes. These occurrences are rare in the dataset and therefore do not affect performance greatly. Natural human speech contains many instances where words are slightly altered to mean different things based on the context they exist in. Examples of this are participles and possessives where the characters in the word change slightly but the meaning varies greatly. These small differences are likely to be confused without information about their context. Testing on a dataset more representative of human speech would likely reveal these flaws in LipNet's methodology.

3 Dataset

The LipNet model is trained and tested on the GRID corpus. This dataset consists of 34 speakers, each giving 1000 sentence utterances. Each sentence is in the form: {command, color, preposition, letter, digit, adverb}. The possible options for each element are:

- Command: BIN, LAY, PLACE, SET
- Color: BLUE, GREEN, RED, WHITE

- Preposition: AT, BY, IN, WITH
- Letter: A - Z + " "
- Number: ZERO - NINE

4 Word Beam Search

Word-level language models provide the grammatical and semantic information that is needed to disambiguate instances where we have words with similar visemes or phonemes. To obtain a word-level language model we employ the use of Word Beam Search in the decoding step.

4.1 Standard Beam Search

Beam search is a graph-based search algorithm that is used to decode the network output to a sentence. The output for each timestep is a tensor of probabilities 28 elements long. Here each character's probability is mapped to one of these outputs. At each timestep, the algorithm adds all of the characters to the existing beams individually. It then evaluates each of these new beams using its heuristic function. The greedy algorithm continues to the next timestep with the top n beams, where n is the beam width, discarding the rest. The heuristic function for the standard beam search implementation is the sum of all probabilities in the beam. In LipNet's case, beam search is augmented to perform the same reduction that the CTC function performs. In the naive model, the output probability for each character is independent of previous letters, meaning the character with the highest probability will always be chosen. This independence means we can forego beam search altogether and simply take the argmax of our output to determine the character that each frame is mapped to. The CTC reduction can then be applied to decode the array into the target string.

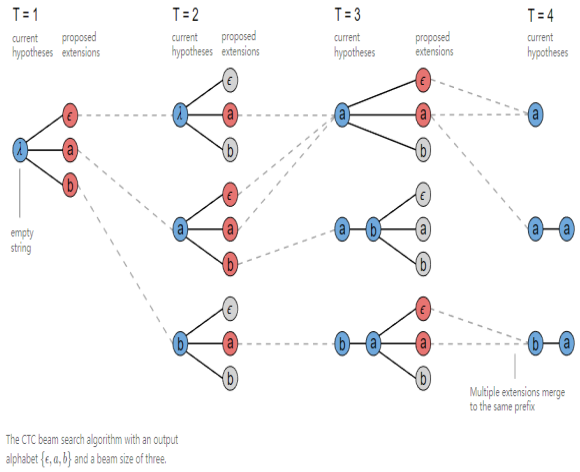

Figure 4. Visual representation of the beam search algorithm

4.2 Word Beam Search

Word Beam Search (WBS), described in [7], employs the same algorithm where beam heuristics are evaluated using the sum of character probabilities across the entire beam. The difference between WBS and standard beam search is that the weights for each beam are supplemented by their viability according to a pre-built word-level language model. The probability for each beam can be increased in three different cases:

1. The character at this timestep completes a word found in the language model's dictionary
2. The character at this timestep completes a word that is part of a common ngram in the dataset when paired with the previously predicted word in the beam.
3. The character can lead to a word that is part of a common ngram in the dataset when paired with the previously predicted word in the beam.

The added probabilities from WBS provide the contextual information lacking in standard beam search. Grammar in natural human speech would be learned through ngrams. In the case that *had to* was previously predicted by a beam, the algorithm could penalize the addition of an extra *o* as it is unlikely that *had too* is common in a dataset adhering to English grammar. Instead, WBS would increase the probability of the beam adding the space character, indicating that the word is complete. WBS could also further increase this probability if "had to" is found to be a common bi-gram in the language model.

5 Our Approach

My approach to solving the issues with LipNet's model is twofold. First, we elevate our model to the word level. The latent character-level language model learned by the network is insufficient to provide the contextual insight between words that is required to disambiguate between words with similar visemes or phonemes. Second, we employ a word-level language model that can supplement the probabilities returned from the original model. This will result in a more robust model that performs better on real-world speech. Both of these steps are accomplished using Word Beam Search. The algorithm uses space characters to split prediction in its beam into distinct words. These words can then be evaluated using the pre-built language model for our dataset. The possible evaluation methods are described below.

5.1 Word Matching

Word matching is the simplest way to add probability to a beam. We restrict predictions to words that appear in the language model's dictionary. At every timestep, the space character is considered for the beam. When testing the space character we also test to see if the sequence of characters preceding it form a word found in the language model's dictionary. If the word exists, the beam is weighted more

heavily. This provides the word-level separation that we require. Each word that is completed is now stored in the beam. We know all of these words are likely to occur in the dataset. Some smoothing is also applied to the language model to not completely remove words that are not present in its dictionary. This is not a problem with the GRID dataset as the dictionary is very small; it is unlikely that a previously unseen word appears in a sample.

This approach could lead to the algorithm favoring short words over long words. If the character at the previous timestep is part of a longer word but also marks the end of a smaller word that is a subset of the larger word, WBS will add weight to the beam and disregard the possibility that the character belongs to a larger word. A likely symptom of this would be the separation of compound words. Words like *database* could easily be separated into *data* and *base*.

5.2 N-gram Matching

Now that we have a separable list of words, we can employ the use of the word-level language model. When the algorithm completes a new word, it can evaluate it against the previously predicted words using the language model. Weight can be added to the beam for every common n-gram that the word appears in.

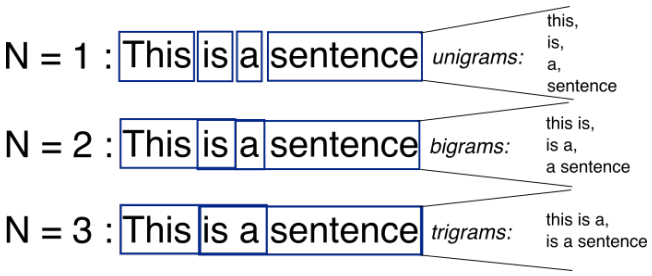

Figure 5. Example of n-grams in a sentence

N-grams provide a sound grammatical context that the model can use. The predicted words should always follow the rules of a specified grammar. The English language has a standard word ordering of Subject, Verb, Object; this ordering is observed during natural human speech. Through common n-grams in the language model, our network is afforded this information. Adding weight when common n-grams are completed helps to ensure predicted words are grammatically correct. By extension, the model also gains semantic insight into its predictions. This added information gives the model the ability to disambiguate between homonyms, as homonyms are not found in similar contexts. From the earlier example of *had to* we can see that though *had too* contains two full words found in the model's dictionary, the pair of them together is likely not a common n-gram.

Alone, this does not solve the problem of the algorithm preferring shorter words. When observing natural English speech, we can see that short words are far more likely to occur than longer words. This distribution implies that shorter words will appear more commonly in n-grams. This will result in WBS adding weight to beams with short words more readily, further pushing the algorithm to favor short words over long words.

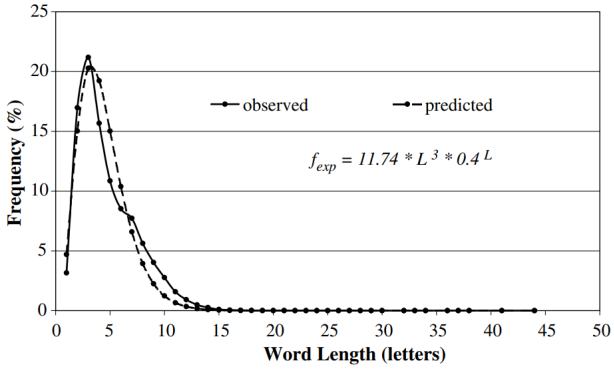

Figure 6. Distribution of words based on word length and frequency in English from [8]

5.3 N-gram Matching with Forecasting

Forecasting is added to WBS to eliminate the network's bias towards short words. Instead of evaluating the language model only when a word is completed, the language model is evaluated at each timestep. WBS will forecast all possible words that can be created using the partial word currently in the beam. It will then evaluate the language model on these words similarly to how the basic n-gram model does. Weights are then added to the beams that could eventually complete a word found in the dictionary. More weight can then be added if the word appears in common n-grams in the language model. The forecasting stops the algorithm from favoring short words by adding weight to all possible word combinations. Words will now only be judged by their semantic and grammatical viability rather than by their length.

This forecasting also leads to a higher number of strong beams. As we cannot pursue all possible beams, we have the highest likelihood of obtaining the correct result if we maximize beam heuristics across all active beams. Forecasting allows the algorithm to prune away character possibilities that will not result in valid words. In contrast to strong beams, beams that can not complete a valid word are weighted poorly. This ensures that they do not appear as possibilities in the next timestep. The result of this pruning will be increased accuracy in decoding as well as higher confidence that the algorithm arrived at the optimal beam.

The increase of accuracy and confidence comes at the expense of computation time. All word possibilities need to be computed at each time step, these candidate words then

Model	Word Error Rate (%)	Character Error Rate (%)
Argmax (No LM)	13.3258	6.7965
Words	12.7362	6.7926
N-Grams	12.7362	6.7926
N-Grams Forecast	12.7195	6.7829
N-Grams Forecast and sampling	12.7195	6.7829

Table 1. Sample test results from the N-gram and Forecast beam evaluation method

need to be evaluated with the language model. The generation and evaluation of these candidates is expensive. To reduce the computational complexity we can predict only a subset of the possible words. This reduces the computational complexity from O(nlogn) to O(n), where n is the size of the dictionary.

6 Testing

We tested the LipNet model augmented with WBS for all four different beam evaluation methods. We evaluate all models on an unseen set of 3986 samples. The word error rate (WER) and character error rate (CER) for each prediction are obtained by getting the Levenshtein distance between the predicted text and the truth text. The WER and CER are averaged across all of the samples. These averages are then compared to the WER and CER of LipNet. We also manually analyze the differences in text predictions to draw inferences about each evaluation method.

7 Results

In this section we discuss the results from testing the language-model-augmented LipNet network with each of the beam evaluation methods in WBS. We describe the effect that each evaluation method has on the model. Further, we attempt to explain where improvements in accuracy or lack of improvements stem from.

7.1 Word Matching

With the word matching beam evaluation method we see an initial decrease in WER. This is due to limiting predicted words to just words that exist in the language model dictionary. As the dictionary for the GRID dataset is small, the language model is able to create a full set of words that can exist in the dataset. This means there will be no words that appear in a sample that do not already exist in the model's dictionary. The increase in word accuracy likely comes from fixing small spelling mistakes in predicted words. Since the model knows all of the words it cannot misspell any of them. We do also see a marginal decrease in CER, again likely resulting from small spelling mistakes in words being fixed.

7.2 N-gram Matching

The N-gram evaluation method is where we would expect to see a marked improvement in accuracy in WER and consequently in CER. This however is not the case; we see no change in accuracy from the more basic word matching evaluation method. This is due to the grammatical structure of the GRID dataset. With such a simple and static grammatical structure, there is little semantic or grammatical information to be gained. The increase from the original LipNet model comes from the same word limiting that takes place in the word matching model.

7.3 N-gram Matching with Forecasting

We see another small improvement in both WER and CER when we include forecasting. This is due to the pruning effect caused by forecasting. Only adding weight to beams that can potentially return valid words ensures we have the strongest beams move forward at each timestep. This leads to more samples returning with their optimal beam. The increase in accuracy is a direct result of this optimization.

7.4 N-gram Matching with Forecasting and Sampling

With the inclusion of sampling we expect to see no improvement. No new evaluation methods are introduced here. Instead, we see that the accuracy in both WER and CER are unchanged. This implies that the performance gains from taking a random sample of possible word predictions in WBS over the standard forecasting method should be accepted. The decrease in the active pruning of beams during WBS does not negatively affect accuracy.

8 Conclusion

The inclusion of a word-level language in LipNet's model seems to have a marginal effect on lipreading accuracy. We do see minor improvements in both WER and CER. This lack of significant improvement may be misleading. Instead of implying that the addition of the language model is ineffective, the lack of increased accuracy could imply that the dataset is insufficient to show the potential gains the language model offers.

We can see instances where the model is exhibiting the semantic understanding that we hoped WBS would afford it. We can see that where the original argmax based model

Prediction	Truth
SET RED BY H EIGHT SOON	SET RED BY H EIGHT SOON
PLACE BLUE AT O SIX NOW	PLACE BLUE AT O SEVEN NOW
LAY BLUE AT N SEVEN NOW	LAY BLUE AT X SEVEN NOW
SET BLUE AT E SIX AGAIN	SET BLUE AT A SIX AGAIN
SET GREEN IN G TWO AGAIN	SET GREEN IN O SEVEN PLEASE
PLACE BLUE IN U NINE PLEASE	PLACE BLUE IN U NINE PLEASE
LAY RED IN U FIVE NOW	LAY RED IN Q FIVE NOW
BIN RED IN X SEVEN PLEASE	BIN RED AT S SEVEN PLEASE
LAY WHITE IN D EIGHT SOON	LAY WHITE AT E EIGHT SOON
PLACE RED BY Q ZERO AGAIN	PLACE RED BY Q ZERO AGAIN

Table 2. Sample test results from the N-gram and Forecast beam evaluation method

predicts *PLACE BLUE AT O SIN NOW*, the language model enhanced network predicts *PLACE BLUE AT O SIX NOW*. Though both predictions are wrong and should read *PLACE BLUE AT O SEVEN NOW*, the language model enhanced network understands that a number should come next. The grammar of this dataset asserts that any bigrams beginning with *O* in this dataset are always followed by a number. If these numbers had any semantic meaning themselves, the network could potentially use that meaning to better predict the sentence.

WBS	LipNet
PLACE BLUE AT O ***SIX*** NOW	PLACE BLUE AT O ***SIN*** NOW

This same understanding acts as a detriment to the model in some cases. We can see that in some cases, the language model enhanced network predicts X where the true character is S. This is the situation where we have two characters with similar visemes we hoped the context from the language model would disambiguate. The problem here is that the letters *X* and *S* have no semantic meaning. Any letter in their place would still be a valid sample in this dataset. In this case there is no meaningful semantic information to be learned to disambiguate the two characters. Instead, WBS will add weight to the beam with the character that appears more often in that specific bigram. In the case of the GRID dataset, this is invalid logic.

WBS	Truth
BIN RED AT ***X*** SEVEN PLEASE	BIN RED AT ***S*** SEVEN PLEASE

To see a marked improvement from the original model, both would have to be trained and tested on a dataset more representative of natural English speech. With a text source bound by complex grammar rules like English, the word-level language model used in WBS would be able to gain the meaningful semantic and grammatical insight it needs to make an effect on lipreading accuracy.

9 Possible Expansions

In this section we discuss possible expansions to this work to further affirm Natural Language Processing's place in automatic speech recognition.

9.1 Dataset Representative of Spoken English

The first step forward is to train and test the different models on a dataset that is more representative of spoken english. As previously discussed, the GRID corpus suffers from a strict grammar that emparts no semantic meaning to the words in each sample. This semantic information missing from the GRID corpus is what a language model relies on to supplement the weights obtained from the character-level LipNet network. Any gain in accuracy is contingent on a dataset that exhibits a grammar where the context of each word is dependent on its surrounding words instead of its position in the sentence.

9.2 Inclusion into Loss Calculation

As it stands, WBS is only applied in the decoding step, after the loss has been calculated. As such, it has no effect on the weights learned in the character-level network. Experimentation should be done to integrate WBS into the loss calculations. This could provide significant improvement in training times. The early stages of training the character-level model are simply the model guessing at characters. The forecasting and word restriction in WBS could offer the model with a good starting foundation.

9.3 Transformers

The transformer architecture has fast been replacing the recurrent neural network. It provides similar functionality to the GRU without any of the long range memory issues the GRU suffers. It is possible that a model based on the transformer architecture could learn both a character-level and word-level language model inherently. This would remove the need to use a pre trained external language model. Experimentation of the viability of the transformer architecture for this task should be performed.

References

[1] Yannis M Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. 2017. LipNet: End-to-End Sentence-level Lipreading. *GPU Technology Conference* (2017). https://github.com/Fengdalu/LipNet-PyTorch

[2] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[3] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. 2006. *The Grid Audio-Visual Speech Corpus.* https://doi.org/10.5281/zenodo.3625687 Collection of this dataset was supported by a grant from the University of Sheffield Research Fund.

[4] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning* 2006, 369–376. https://doi.org/10.1145/1143844.1143891

[5] Awni Hannun. 2017. Sequence Modeling with CTC. *Distill* (2017). https://doi.org/10.23915/distill.00008 https://distill.pub/2017/ctc.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25 (01 2012). https://doi.org/10.1145/3065386

[7] H. Scheidl, S. Fiel, and R. Sablatnig. 2018. Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm. In *16th International Conference on Frontiers in Handwriting Recognition*. IEEE, 253–258.

[8] Bengt Sigurd, Mats Eeg-Olofsson, and Joost van de Weijer. 2004. Word length, sentence length and frequency – Zipf revisited. *Studia Linguistica* 58 (04 2004), 37 – 52. https://doi.org/10.1111/j.0039-3193.2004.00109.x

A Full Results

In this appendix we show the full results from testing each WBS beam evaluation method

Prediction	Truth
SET RED BY H EIGHT SOON	SET RED BY H EIGHT SOON
PLACE BLUE AT O SIN NOW	PLACE BLUE AT O SEVEN NOW
LAY BLUE AT N SEVEN NOW	LAY BLUE AT X SEVEN NOW
SET BLUE AT E SIX AGAIN	SET BLUE AT A SIX AGAIN
SET GREEN IN G TWO AGAIN	SET GREEN IN O SEVEN PLEASE
PLACE BLUE IN U NINE PLEASE	PLACE BLUE IN U NINE PLEASE
LAY RED IN U FIVE NOW	LAY RED IN Q FIVE NOW
BIN RED IN X SEVEN PLEASE	BIN RED AT S SEVEN PLEASE
LAY WHITE IN D EIGHT SOON	LAY WHITE AT E EIGHT SOON
PLACE RED BY Q ZERO AGAIN	PLACE RED BY Q ZERO AGAIN

Table 3. Argmax

Prediction	Truth
SET RED BY H EIGHT SOON	SET RED BY H EIGHT SOON
PLACE BLUE AT O SIX NOW	PLACE BLUE AT O SEVEN NOW
LAY BLUE AT N SEVEN NOW	LAY BLUE AT X SEVEN NOW
SET BLUE AT E SIX AGAIN	SET BLUE AT A SIX AGAIN
SET GREEN IN G TWO AGAIN	SET GREEN IN O SEVEN PLEASE
PLACE BLUE IN U NINE PLEASE	PLACE BLUE IN U NINE PLEASE
LAY RED IN U FIVE NOW	LAY RED IN Q FIVE NOW
BIN RED IN X SEVEN PLEASE	BIN RED AT S SEVEN PLEASE
LAY WHITE IN D EIGHT SOON	LAY WHITE AT E EIGHT SOON
PLACE RED BY Q ZERO AGAIN	PLACE RED BY Q ZERO AGAIN

Table 4. N-gram

Prediction	Truth
SET RED BY H EIGHT SOON	SET RED BY H EIGHT SOON
PLACE BLUE AT O SIX NOW	PLACE BLUE AT O SEVEN NOW
LAY BLUE AT N SEVEN NOW	LAY BLUE AT X SEVEN NOW
SET BLUE AT E SIX AGAIN	SET BLUE AT A SIX AGAIN
SET GREEN IN G TWO AGAIN	SET GREEN IN O SEVEN PLEASE
PLACE BLUE IN U NINE PLEASE	PLACE BLUE IN U NINE PLEASE
LAY RED IN U FIVE NOW	LAY RED IN Q FIVE NOW
BIN RED IN X SEVEN PLEASE	BIN RED AT S SEVEN PLEASE
LAY WHITE IN D EIGHT SOON	LAY WHITE AT E EIGHT SOON
PLACE RED BY Q ZERO AGAIN	PLACE RED BY Q ZERO AGAIN

Table 5. N-gram and Forecast

Prediction	Truth
SET RED BY H EIGHT SOON	SET RED BY H EIGHT SOON
PLACE BLUE AT O SIX NOW	PLACE BLUE AT O SEVEN NOW
LAY BLUE AT N SEVEN NOW	LAY BLUE AT X SEVEN NOW
SET BLUE AT E SIX AGAIN	SET BLUE AT A SIX AGAIN
SET GREEN IN G TWO AGAIN	SET GREEN IN O SEVEN PLEASE
PLACE BLUE IN U NINE PLEASE	PLACE BLUE IN U NINE PLEASE
LAY RED IN U FIVE NOW	LAY RED IN Q FIVE NOW
BIN RED IN X SEVEN PLEASE	BIN RED AT S SEVEN PLEASE
LAY WHITE IN D EIGHT SOON	LAY WHITE AT E EIGHT SOON
PLACE RED BY Q ZERO AGAIN	PLACE RED BY Q ZERO AGAIN

Table 6. N-gram and Forecast with Sampling