Proceedings of the Seminar

Deep Learning

University of Colorado, Colorado Springs

August 5, 2022

Editors: Jugal K. Kalita, Oluwatosin Oluwadare and Adham Atyabi

Funded by

National Science Foundation

Preface

It is with great pleasure that we present to you the papers describing the research performed by the NSF-funded Research Experience for Undergraduates (REU) students, who spent 10 weeks during the summer of 2022 at the University of Colorado, Colorado Springs. Within a very short period of time, the students were able to choose cutting-edge projects involving machine learning in the areas of natural language processing, bioinformatics and computational medicine; write proposals; design interesting algorithms and approaches; develop code; and write scholarly papers describing their findings. We hope that the students will continue working on these projects and submit papers to conferences and journals within the next few months. We also hope that it is the beginning of a fruitful career in research and innovation for all our participants.

We thank the National Science Foundation for funding our REU site. We also thank the University of Colorado, Colorado Springs, for providing an intellectually stimulating environment for research. In particular, we thank Dr. Terrance Boult, who was a helpful and stimulating mentor for the REU students. We also thank Sharon Huscher for working out all the financial and administrative details. We thank Dr. Donald Rabern, the Dean of the College of Engineering and Applied Science, and Dr. Thottam Kalkur, the Chair of the Electrical and Computer Engineering Department for their support. We also thank our graduate students, in particular, Ali Al Shami, Dino Bonaldo, Uma Chinta, Steve Cruz, Timothy Flink, Kaden Griffith, Justin Leo, Thomas Matthews, Steven Paligo, Wesley Robbins, Yousra Shleibik, and Zanyar Zohourianshahzadi for helping the students with ideas as well as with presentations on some of the lastest papers, and systems and programming issues. Our gratitude to Ginger Boult for being the "REU Mom" and having the welfare of the REU interns at her heart all through the summer. Special thanks to Parker Hicks (UCCS REU 2021) of University of Colorado---Denver Anschutz Medical Campus, Van Hovenga (UCCS BS 2020, MS 2022, heading to University of California, Berkeley in Fall 2022 for PhD) and Abigail Swenor (UCCS BS 2022, UCCS REU 2021) of Notre Dame University, for taking part in panel discussions on how to apply to graduate school.

Sincerely,

Jugal Kalita
jkalita@uccs.edu
Professor

Oluwatosin Oluwadare
ooluwada@uccs.edu
Assistant Professor

Adham Atyabi
aatyabi@uccs.edu
Assistant Professor

August 5, 2022

Table of Contents

NSF REU Seminar on Machine Learning

Department of Computer Science

University of Colorado, Colorado Springs

Engineering 105

August 5, 2022: Friday

10*:00-10:10 AM:* Welcome Remarks by *Dr. Thottam Kalkur*, Professor and Chair, Electrical and Computer Engineering, University of Colorado, Colorado Springs

10:10-*11:25 AM* Session Chair*: Dr. Terrance Boult,* El Pomar Endowed Chair of Innovation and Security and Professor of Computer Science, University of Colorado, Colorado Springs, CO

- 10:10-10:35 *Abigail Newcomb,* St. Olaf College, Northfield, MN: <u>Explaining Math Word Problem Solvers</u>
- 10:35-11:00 G*abriel Mantione-Holmes*, Lewis and Clark College, Portland, OR: <u>Utilizing priming to identify optimal class ordering to alleviate the problems of catastrophic forgetting</u>
- 11:00-11:25 *Daniel DeGenaro,* University of Massachusetts, Amherst, MA: <u>CAMeMBERT: Cascading Assistant-Mediated Multilingual BERT</u>

<u>11:25-11:35 Break</u>

11:35-12:25 PM Session Chair: *Dr. Adham Atyabi*, Assistant Professor of Computer Science, University of Colorado, Colorado Springs, CO

- 11:35-12:00 *Aaron Serianni*, Princeton University, Princeton, NJ: <u>Training-free Neural Architecture Search for RNN and Transformer Architectures</u>
- 12:00-12:25 *Raymond Dueñas*, Stanislaus State University, Turlock, CA: <u>Light Weight Transformers Ramp up Autonomy of UAVs</u>

12:25-1:25 PM: Lunch

12:25-1:25 PM: Welcome Back Remarks

1:25-2:40 PM Session Chair: *Dr. Oluwatosin Oluwadare*, Assistant Professor of Computer Science, University of Colorado, Colorado Springs, CO

- 1:25-1:50 *Zachary Snow*, University of Kentucky, Lexington, KY: <u>The Effects of Subject Transfer on Transformer-Based EEG Classification of Finger Movement</u>
- *1*:50-2:15 *Marcin Pawlukiewicz,* University of Rhode Island, Kingston, RI: <u>3DChromoTwist: Development of a 3D Chromosome Structure Reconstruction Game for Educational Purposes</u>
- *2*:15-2:40 *Nicole Baugh,* North Carolina State University, Raleigh, NC*:* <u>RECSplice: Splice Site Prediction Using Recurrent Neural Networks</u>

<u>2</u>:40 PM: Closing Remarks by Drs. Oluwatosin Oluwadare, Adham Atyabi and Jugal Kalita

Our Session Chairs and Guests

Dr. Adham Atyabi is an Assistant Professor in the Department of Computer Science, University of Colorado, Colorado Springs. His expertise are in cognitive and computational neuroscience, brain-computer interface, image and signal processing, and deep learning. At UCCS for 4 years, he obtained his PhD from Flinders University, Australia, and held post-doc positions at Yale and University of Washington. Dr. Atyabi has published 70 papers in various conferences and journals.

Dr. Terrance E. Boult is El Pomar Endowed Chair of Innovation and Security and Professor of Computer Science at the University of Colorado Colorado Springs. Dr. Boult was the visionary behind the unique Bachelor of Innovation Family of degrees, and actively teaches mostly in that program. Dr. Boult runs the Vision and Security Technology Lab (VAST lab) where his students and he are doing projects in Security including machine learning, computer vision, surveillance, and biometrics. Dr. Boult works with The El Pomar Institute for Innovation and Commercialization through which he works with many local and student companies.

Dr. Thottam Kalkur is Professor and Chair of the Electrical and Computer Engineering Department at the University of Colorado, Colorado Springs. His areas of research are microelectronics circuit design, device physics, ferroelectrics for tunable RF circuit applications, nano-crystalline memories, polarization switching data converters, deep submicron device and circuit modeling, MEMS based sensors and switches, and radiation hardened circuit design.

Dr. Patrick McGuire is an Associate Professor in the Department of Curriculum and Instruction focusing on STEM education. He is also the Chair of the department. Dr. McGuire also serves as the College of Education Co-Director of UCCSTeach, an inquiry-based program designed to prepare the next generation of secondary mathematics and science teachers (see www.uccs.edu/uccsteach for more information). His research interests lie in the intersection of curriculum, instructional technology and STEM education. Before joining the UCCS faculty in 2010, Pat worked as a high school mathematics teacher in Pittsburgh, PA and as a researcher at Carnegie Mellon University. Dr. McGuire is the evaluator for the REU grant.

Dr. Oluwatosin Oluwadare is an Assistant Professor of Computer Science and Innovation at the University of Colorado, Colorado Springs. Dr. Oluwadare's research focus areas are Bioinformatics and Computational Biology, Machine Learning and Data Mining, and Deep Learning and Reinforcement Learning. He led the development of a software app called EyeCYou (http://eyecyouapp.com/) that uses AI to provide the facial description of a person to the visually impaired .

NSF REU Midsummer Presentation Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 103
Monday, July 11 and Wednesday, July 13, 2022

11:30-1:00 PM, July 11
Session Chair*: Justin Leo,* PhD Student, Department of Computer Science, University of Colorado, Colorado Springs.

Aaron Serianni, Princeton University, Princeton, NJ: Training-free Neural Architecture Search for RNN and Transformer Architectures

Raymond Dueñas, Stanislaus State University, Turlock, CA: Light Weight Transformers Ramp up Autonomy of UAVs

Marcin Pawlukiewicz, University of University of Rhode Island, Kingston, RI: 3DChromoTwist: Development of a 3D Chromosome Structure Reconstruction Game for Educational Purposes

Zachary Snow, University of Kentucky, Lexington, KY: The Effects of Subject Transfer on Transformer-Based EEG Classification of Finger Movement

11:30-1:00 PM, July 13
Session Chair: Yousra Shleibik, MS Student, Department of Computer Science, University of Colorado, Colorado Springs, CO

Gabriel Mantione-Holmes, Lewis and Clark College, Portland, OR: Utilizing priming to identify optimal class ordering to alleviate the problems of catastrophic forgetting

Daniel Degenaro, University of Massachusetts, Amherst, MA: CAMeMBERT: Cascading Assistant-Mediated Multilingual BERT

Abigail Newcomb, St. Olaf College, Northfield, MN: Explaining Math Word Problem Solvers

Nicole Baugh, North Carolina State University, Raleigh, NC: RECSplice: Splice Site Prediction Using Recurrent Neural Networks

Our Session Chairs

Justin Leo is a PhD student in the Department of Computer Science at the University of Colorado, Colorado Springs (UCCS). He participated in the REU program on Machine Learning at UCCS in 2019. He published a conference paper based on his REU work in 2020, followed by a recent paper on incremental class learning in IEEE Transactions on Neural Networks and Learning Systems.

Yousra Shleibik is an MS student in the Department of Computer Science at the University of Colorado, Colorado Springs (UCCS). She is a Fulbright Scholar from Libya. Her interests are in applying Deep Learning techniques in Artificial and Virtual Reality.

NSF REU Proposal Presentation Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 101
Friday, June 10, 2022

1:45-1:50 PM: Welcome Remarks by Dr. Donald Rabern, Dean, College of Engineering and Applied Science, University of Colorado, Colorado Springs

1:55-2:55 PM
Session Chair: Zanyar Zohoruianshahzadi, PhD student, Department of Computer Science, University of Colorado, Colorado Springs, CO

Aaron Serianni, Princeton University, Princeton, NJ: Training-free Neural Architecture Search for NLP RNNs and Transformers

Abigail Newcomb, St. Olaf College, Northfield, MN: Probing Math Word Problem Solvers

Daniel DeGenaro, University of Massachusetts, Amhert, MA: CAMeMBERT: Cascading Assistant-Mediated Multilingual BERT

3:05-4:05 PM
Session Chair: Steve Cruz, Department of Computer Science, University of Colorado, Colorado Springs, CO

Gabriel Mantione-Holmes, Lewis and Clark College, Portland, OR: Utilizing Random Perturbations to Alleviate Catastrophic Forgetting in Lifelong Learning

Marcin Pawlukiewicz, University of University of Rhode Island, Kingston, RI: Development of a 3D Structure Reconstruction Game for Educational Purposes

Nicole Baugh, North Carolina State University, Raleigh, NC*:* RECSplice: Splice Site Prediction Using Recurrent Neural Networks

4:15-4:55 PM
Session Chair: Uma Chinta, PhD student, Department of Computer Science, University of Colorado, Colorado Springs, CO

Raymond Dueñas, Stanislaus State University, Turlock, CA: Light Weight Transforms Ramp up Autonomy of UAV's

Zachary Snow, University of Kentucky, Lexington, KY: CNN and Transformer-Based EEG and ECoG Classification of Finger Movement

Our Session Chairs

Uma Chinta is a fifth year PhD student at the University of Colorado, Colorado Springs. She works in and manages the Neurocognition Lab. Her research interests are in facial expression analysis, sentiment analysis and multimodal emotion recognition. She has submitted multiple papers, which are under review. She is planning to work on data collection for her studies involving autistic kids, starting Fall 2022.

Zanyar Zohourianshahzadi is a Ph.D. candidate at UCCS. His research mainly focuses on image captioning and multimodal learning. His latest work, titled "Neural Attention for Image Captioning: Review of Outstanding Methods" explains the evolution path of attention mechanisms in the context of image captioning. He has also published "Neural Twins Talk" at IEEE nternational Conference on Humanized Computing and Communication with Artificial Intelligence (HCCAI) and a followup work titled "Neural Twins Talk and Alternative Calculations" in the International Journal of Semantic Computing.

Steve Cruz is a PhD student at UCCS, graduating in Summer 2022. His primary area of research is open-set recognition. He has published papers at conferences such as IEEE Computer Vision and Pattern Recognition (CVPR), Association for the Advancement of Artificial Intelligence (AAAI), and IEEE Winter Conference on Applications of Computer Vision (WACV). He also serves as a reviewer for these conferences.

Explaining Math Word Problem Solvers

Abby Newcomb
St. Olaf College
1500 St. Olaf Ave
Northfield, Minnesota 55057
abbynewcomb13@gmail.com

Jugal Kalita
University of Colorado, Colorado Springs
420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
jkalita@uccs.edu

Abstract

Automated math word problem solvers based on neural networks have successfully managed to obtain 70-80% accuracy in solving arithmetic word problems. However, it has been shown that these solvers may rely on superficial patterns to obtain their equations. In order to determine what information math word problem solvers use to generate solutions, we remove parts of the input and measure the model's performance on the perturbed dataset. Our results show that the model is not sensitive to the removal of many words from the input and can still manage to find a correct answer when given a nonsense question. This indicates that automatic solvers do not follow the semantic logic of math word problems, and may be overfitting to the presence of specific words.

1 Introduction

Math word problem (MWP) solving is an area of natural language processing (NLP) that uses machine learning to solve simple arithmetic problems. MWPs consist of a few sentences of text including a few numbers and an unknown quantity, similar to problems humans are presented with in grade school. Neural networks are trained to generate the correct equation which computes the unknown quantity. Little is known about *how* these models manage to solve math word problems. In this paper, we remove parts of math word problems and measure the model's performance on the changed data in order to ascertain which words the model is using to choose the correct equation.

Various parts of speech work together to construct the full meaning of a sentence, so even when a certain part of speech is removed, other words may still indicate the desired operation. In order to more specifically gauge which words are important to the model's prediction, we employ input reduction, a strategy that iteratively removes the least important word from the input until the model produces an incorrect result. This method allows us to see how removing specific words affects the model.

We also perform analysis of which words appear most frequently in the datasets used to train the model. We also look at the most common words for each type of problem (+, -, *, /, multiple) to see whether certain words appear to indicate specific operations.

In order to determine which parts of speech are most important to MWP solvers, we remove specific words from MWP test datasets and test a Seq2seq MWP solver on its ability to determine the correct answer on these reduced problems. The contributions of this paper are as follows:

- We show that the lexical diversity of MaWPS is low.
- We show that the RNN Seq2seq solver performs little semantic reasoning, since it can produce correct answers with significantly reduced input.

We begin by explaining related work, then cover the methods and results of each experiment in turn, followed by the conclusion.

2 Related Work

Various neural network MWP solvers have been created and benchmarked on well-known datasets. Few explainability techniques have yet been applied to MWP solving.

Math Word Problems

The current most prolific datasets for Math Word Problem solving are MaWPS (Koncel-Kedziorski et al. 2016) and ASDiv-A (Miao, Liang, and Su 2021). These datasets are currently the largest ones available, though they are quite small for machine learning datasets. MaWPS has 2373 MWPs while ASDiv-A has only 1218 problems.

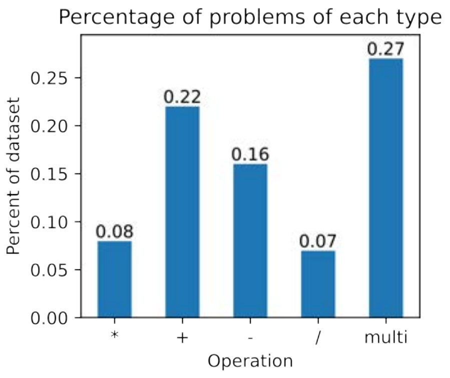


Figure 1: Percentage of MWPs in MaWPS dataset of each operation type, on average across all CV folds.

Various types of neural networks for solving MWPs have been developed. Wang, Liu, and Shi (2017) use a GRU encoder and an LSTM decoder in a sequence to sequence approach. Another model is a graph to tree model proposed by Zhang et al. (2020), which uses a graph transformer and tree structured decoder to generate the MWP solution expression tree. Griffith and Kalita (2019) use a transformer-based model. Xie and Sun (2019) use a model called GTS in a process they call goal decomposition to find relationships between quantities. Their approach uses feed-forward networks and an RNN model at different steps in the algorithm.

Though these models obtain high accuracy, their success was called into question when MWP solvers were shown to obtain similar accuracy when the actual question was removed, leaving only the descriptive body of text at the beginning of the problem (Patel, Bhattamishra, and Goyal 2021). MWP solvers also perform poorly on the SVAMP challenge dataset, which was specifically generated to require attention to the question itself (Patel, Bhattamishra, and Goyal 2021). This implies that the solvers are relying on superficial patterns in the initial text rather than actually answering the question posed in the problem. However, it was later shown that performance on the SVAMP dataset could be improved simply by generating more data to increase the size of MWP training datasets (Kumar, Maheshwary, and Pudi 2022).

Explainability Techniques

The strategy of removing parts of the input to an NLP model is often used to explain a model's decisions. Importance scores have been assigned to words in the input by looking at the effects of removing those words (Li, Monroe, and Jurafsky 2017). Similarly, the process of input reduction involves successively removing the word that affects the model's confidence score the least, until we are left with the smallest possible input with which the model can still make a correct prediction (Feng et al. 2018). This process shows us which words in the input are most important to the model's prediction. These methods, among others, have been implemented by Wallace et al. (2019) in their AllenNLP framework for NLP explainability techniques. However, applications of these methods often focus on large models such as BERT and tasks such as to sentiment analysis, reading comprehension, or textual entailment. This method has not yet been applied to MWP solving.

Another method of understanding NLP model predictions is adversarial attacks, in which various changes are made to the input of a model, and the performance of the model is measured in order to determine how sensitive the model is to the changes in the perturbed dataset. Adversarial attacks are different from the aforementioned methods because the new inputs to the model are meant to be semantically equivalent to the previous inputs and should still be grammatically correct (Lee, Kim, and Hwang 2019). Adversarial examples have been used for interpretability of reading comprehension systems (Jia and Liang 2017) and question answering systems (Lee, Kim, and Hwang 2019) in the past. Adversarial attacks involving question reordering and sentence paraphrasing were also used by Kumar, Maheshwary, and Pudi (2021) to show that MWP solvers are not robust to these seemingly irrelevant perturbations.

3 Problem Statement

The question remains to what degree MWP solvers perform semantic reasoning, and what information they use to generate an equation for a solution to a given problem. We apply various methods to search for trigger words and other superficial patterns that the model may be relying on instead of semantic reasoning.

4 Experiment 1: Removing Parts of Speech

We removed various parts of speech from the MWPs and tested an MWP solver's performance on the perturbed datasets in order to see how important different types of words are to the model. A large decrease in accuracy due to the removal of a part of speech indicates that that part of speech is important to the model's prediction, since the model cannot perform as well without it.

Methods

We generate perturbed MWPs by identifying parts of speech using the Natural Language Toolkit (NLTK) part-of-speech tagger (Bird, Klein, and Loper 2009) and then removing the targeted words. We use the Seq2seq model created by Patel, Bhattamishra, and Goyal (2021) for all experiments. We also use Patel, Bhattamishra, and Goyal (2021)'s optimized parameters for training. Two models were trained on either MaWPS or AsDIV-A with 5 fold cross-validation, and then each was evaluated on perturbed examples from its respective dataset. Accuracy is measured on the model's success in generating the correct answer, rather than by the proximity of the generated equation to the true equation.

As a bit of preliminary analysis, we looked at the relative concentration of different types of MWPs in MaWPS, as seen in Figure 1. The first four categories are characterized by having a single operation of the specified type, while the problems in the "multi" category have multiple operations of different types in them. The majority of problems in MaWPS (73%) have only one operation. The dataset appears to represent addition and subtraction the best, and have a much smaller number of multiplication and division problems. This may contribute to the slightly decreased accuracy on multiplication and division problems visible on Figure 6 in the Appendix.

Results

The models' accuracy on each perturbed dataset is listed in Figures 2 and 3, and Table 3 shows all percent accuracies and decreases in accuracy. On the original dataset, the model trained on ASDiv-A had 72.4% accuracy while the MaWPS model had 86.5% accuracy. Removal of common adjectives such as "more" resulted in accuracy decreases of 5.4% and 2.4% respectively, while removing question adjectives such as "how" decreased accuracy by only 2.1% and 1.3%, and removal of all adjectives decreased accuracy by 5.1% and 2.9%. Removal of named entities such as "Jim" was only conducted with MaWPS because of the formatting of the data. MaWPS model accuracy decreased by only

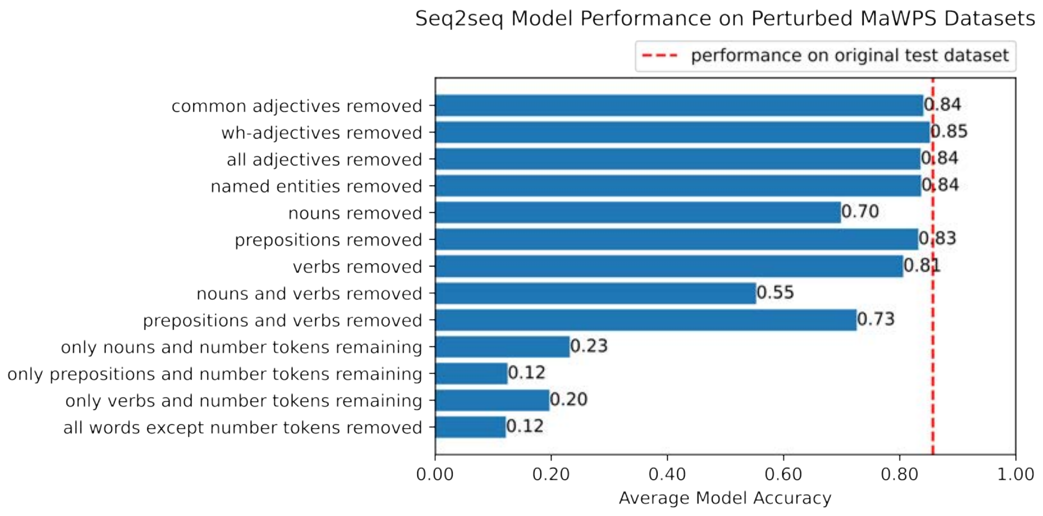


Figure 2: The average accuracy of the Seq2seq model trained on the MaWPS dataset, when evaluated on various perturbed datasets. The model's average accuracy on the original test dataset is indicated by the red dashed line.

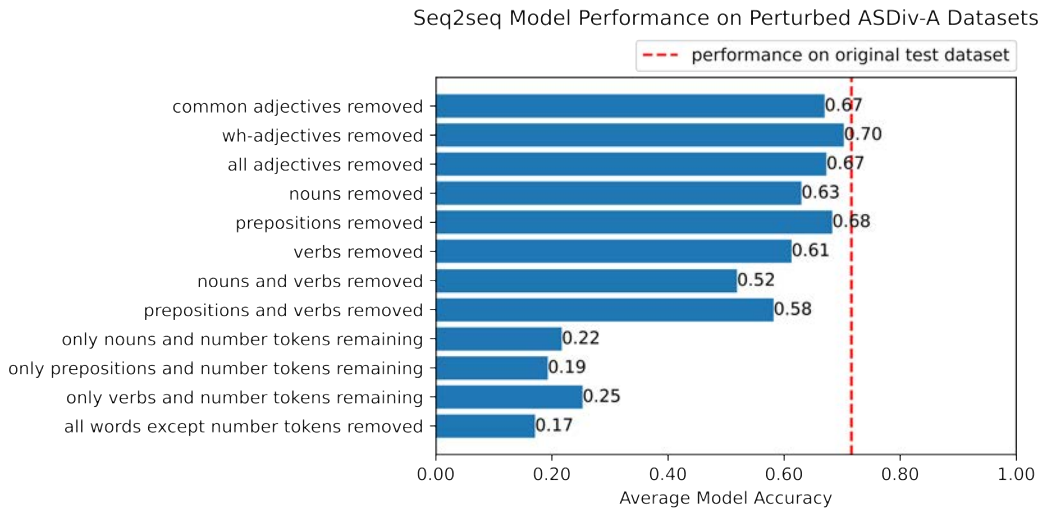


Figure 3: The average accuracy of the Seq2seq model trained on the ASDiv-A dataset, when evaluated on various perturbed datasets. The model's average accuracy on the original test dataset is indicated by the red dashed line.

	Count	Pct
book	68	0.16
will	62	0.14
were	61	0.14
box	52	0.12
tree	52	0.12
total	51	0.12
at	49	0.11
is	49	0.11
pick	48	0.11
from	46	0.11
park	45	0.1

(a) Addition

	Count	Pct
dollar	63	0.19
total	52	0.16
game	44	0.13
balloon	43	0.13
book	41	0.12
will	40	0.12
at	39	0.12
were	39	0.12
pick	37	0.11
is	37	0.11
all	36	0.11

(b) Subtraction

	Count	Pct
card	30	0.18
were	26	0.16
box	25	0.15
will	25	0.15
now	23	0.14
total	23	0.14
book	22	0.13
from	22	0.13
pick	21	0.13
one	21	0.13
all	20	0.12

(c) Multiplication

	Count	Pct
piece	23	0.16
his	23	0.16
dollar	21	0.15
box	21	0.15
from	19	0.14
make	18	0.13
hour	18	0.13
at	17	0.12
now	16	0.11
balloon	16	0.11
game	15	0.11

(d) Division

	Count	Pct
dollar	89	0.16
box	77	0.14
piece	73	0.13
book	69	0.13
total	69	0.13
at	65	0.12
will	63	0.11
all	61	0.11
game	61	0.11
from	61	0.11
would	61	0.11

(e) Multiple

Table 1: The top words for each operation in MaWPS CV Fold 1, excluding words that appeared in all 5 lists, by count of MWPs it appears in. Percentage of MWPs of that operation that the word appears in is also provided for comparison's sake.

2.8% with no named entities. Removal of all nouns, including named entities and all common nouns, decreased accuracy by 9.4% on ASDiv-A and 16.6% on MaWPS. Removing prepositions decreased accuracy by 4.1% and 3.3% respectively. Removing verbs decreased accuracy by 11.1% in the ASDiv-A model and by 5.9% on the MaWPS model.

We also tested the models on datasets with two different parts of speech missing. On a dataset with all nouns and verbs missing, the ASDiv-A model accuracy decreased by 20.5% and MaWPS by 31.2%. With all prepositions and verbs removed, the models' accuracy decreased by 14.2% and 13.9% respectively.

The model was also tested on datasets where only a specific part of speech and the number tokens were left in the MWP, with all other words removed from the input. The results on these datasets tended to somewhat mirror the model's performance on the datasets with that part of speech removed.

On a dataset with all words except for the number tokens removed, the model achieved 12.2% accuracy for MaWPS and 17.1% accuracy on AsDIV-A. It is difficult to calculate what a completely random accuracy would be and how close these are to random guesses because of the complexity of multiple operations, but the AsDIV-A model does manage a significantly higher accuracy, which indicates that it may not rely on the word content as much as the MaWPS model.

Discussion

The model's overall higher accuracies on the MaWPS dataset can likely be attributed to its size, since with 2373 MWPs it is nearly twice as large as ASDiv-A's 1218 problems. The MaWPS model was also less affected by the removal of any single part of speech compared to the ASDiv-A model (average accuracy difference of 5.0% to ASDiv-A's 6.2%), and thus seems to be less sensitive to this type of perturbation overall. The MaWPS model was also more affected by the removal of multiple parts of speech, as the decrease in performance on the twice perturbed datasets was larger than the sum of the decrease in performance on either of the once perturbed datasets, which was not the case for the ASDiv-A model.

The removal of any single part of speech does not appear to significantly affect either model. Overall, the MaWPS model was most affected by the removal of nouns at a 16.6% decrease in accuracy, and the ASDiv-A model was most affected by the removal of verbs at an 11.1% decrease. As hypothesized, certain operations are more affected by the removal of some parts of speech more than others, as seen in Figures 6 and 5 in the Appendix. The models' decent performance on these reduced datasets indicates that no single part of speech is incredibly important to its decision.

However, both models were still achieving an accuracy above 50% with no nouns or verbs in the MWPs. This relatively high accuracy indicates that these models are likely not performing semantic reasoning about the events described in the MWP, since there is not enough information in the problem with no verbs or nouns for the model to truly be reasoning about the quantities present. Instead, the solver may be relying on the presence of trigger words. For example, the words "more" and "together" are likely to signal addition even if the model is given no additional context, while "each" may signal multiplication or division.

For the datasets with only one part of speech and the number tokens remaining, no extremely large jumps in accuracy were observed that would suggest that the model relies entirely on one part of speech. However, accuracy was nearly doubled from 12% to 23% with only nouns in the MaWPS model, which does suggest at least some reliance on the presence of certain nouns in this model since clearly with only nouns to go draw its conclusions, no logical reasoning of events is possible.

5 Experiment 2: MaWPS Word Frequency

In this experiment, we examine the diversity, or lack thereof, of words in the MaWPS' dataset's vocabulary. Our work is intended to reveal possible trigger words that may frequently appear in some types of problems but not others.

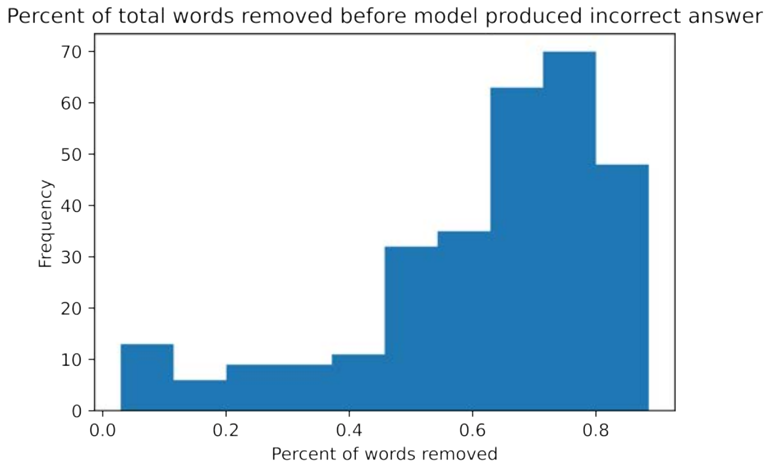


Figure 4: A histogram of the percentage of words in a given MWP were removed before the model produced an incorrect solution to the problem. This histogram does not include MWPs that the model gave an incorrect prediction with the original text.

Methods

We looked at the word frequency of words in the first cross-validation fold of the MaWPS dataset, both the training and testing datasets. The problem texts were first set to all lower-case letters, then stemmed and lemmatized in order to count all occurrences of the words.

We counted the number of MWPs that each word appeared in rather than the total number of appearances of each word. We found the top 50 words, by number of MWPs the word appeared in, for every operation type (+, -, *, /, multiple), then filtered out any words that appeared in every list. In this way we can see which words are uniquely frequent in specific operations, and are not just frequent in the corpus overall.

Results

The results are shown in Table 1. We can see that these words often appear in 10-20% of all problems of a given type, though the majority of the words do not appear to have any correlation to the type of operation that they most often appear in.

Discussion

None of the most popular words appeared to be relevant to the category of problem that they most frequently appeared in. The fact that these words are appearing so frequently indicates a low lexical diversity in the MaWPS dataset, which may encourage the model to rely on the occurrence of these words to classify problems into different operations.

6 Experiment 3: Input Reduction

We used input reduction to uncover how many words can be removed from an MWP before the model will produce an incorrect answer. If very few words remain and have little to do with the correct equation, it suggests that the model is not performing much semantic reasoning between quantities in order to find the correct equation.

Methods

Our approach is based on the work of Feng et al. (2018), but does not follow their exact methodology. We implemented confidence scores using the posterior probability of each label, summed those probabilities and divided by the number of outputs, since we were using an RNN model. For the input reduction process, we iteratively removed the word which reduced the model's confidence score the least.

We used only the RNN Seq2seq model created from the first CV fold of MaWPS for our input reduction predictions.

Results

A histogram of the percentage of words removed when the model gave an incorrect prediction is shown in Figure 4. The histogram does not include MWPs that the model got wrong with the original text. The mean percentage of words removed is 62.3%, while the median is 68.1%. This means that the model produces the correct prediction with less than 68.1% of the words for half of the problems it is able to solve.

An example of the input reduction process is shown in Figure 4 in the Appendix. In this example, 22 words are removed before the model produces an incorrect equation. The most reduced input to receive a correct equation is "his number0 each his number1 many," which arguably contains little to no information about what the correct equation is, and yet the model still solves the problem with high confidence (99%).

Discussion

The results of the input reduction experiment show that in most cases more than half of the total words can be removed from the MWP before the model produces an incorrect answer. With over half of the words removed, these problems are nonsensical to humans, as in Table 4. This indicates that the model is not truly performing reasoning about the sequence of events explained in the problem, since it can still produce a correct equation with over half of the information removed from the input.

7 Future Work

We would like to implement the gradient-based method used by Feng et al. (2018) in order to obtain a more objective idea of how much removing a given word affects the model. The current confidence score approach produces very high confidence on almost every input, even when it is wrong, which reduces the credibility of our input reduction results.

We would also like to implement the high-entropy output fine-tuning suggested by Feng et al. (2018) to possibly improve the interpretability and accuracy of the RNN Seq2seq MWP solver.

Another possible avenue of word would be to increase the lexical diversity of MaWPS by writing code to change words to synonyms before the MWPs are fed into the model for training. This way, the model would not be able to rely on the high frequency of certain words to make its predictions.

8 Conclusion

The results of Experiment 1, parts of speech removal, indicated a small reliance on some parts of speech, especially nouns and verbs. The AsDIV-A model was also shown to be more reliant on specific parts of speech than MaWPS, perhaps indicating some overfitting to those words. Experiment 2, word frequency in MaWPS, shows that the lexical diversity of MaWPS is low. Experiment 3, input reduction, shows that well over half of the words in a given MWP can be removed before the model gives an incorrect prediction. This shows that the model is not using all of the information in the question to make its prediction, and may be relying on occurrences of some of the words from Experiment 2, or some other superficial patterns, to make its predictions.

9 Acknowledgements

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* O'Reilly Media, Inc.

Feng, S.; Wallace, E.; Grissom II, A.; Iyyer, M.; Rodriguez, P.; and Boyd-Graber, J. 2018. Pathologies of Neural Models Make Interpretations Difficult. arXiv:1804.07781.

Griffith, K., and Kalita, J. 2019. Solving Arithmetic Word Problems Automatically Using Transformer and Unambiguous Representations. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 526–532.

Jia, R., and Liang, P. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. arXiv:1707.07328.

Koncel-Kedziorski, R.; Roy, S.; Amini, A.; Kushman, N.; and Hajishirzi, H. 2016. MAWPS: A Math Word Problem Repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1152–1157. San Diego, California: Association for Computational Linguistics.

Kumar, V.; Maheshwary, R.; and Pudi, V. 2021. Adversarial Examples for Evaluating Math Word Problem Solvers. Technical Report arXiv:2109.05925, arXiv. arXiv:2109.05925.

Kumar, V.; Maheshwary, R.; and Pudi, V. 2022. Practice Makes a Solver Perfect: Data Augmentation for Math Word Problem Solvers. Technical Report arXiv:2205.00177, arXiv.

Lee, G.; Kim, S.; and Hwang, S.-w. 2019. QADiver: Interactive Framework for Diagnosing QA Models. *Proceedings of the AAAI Conference on Artificial Intelligence* 33(01):9861–9862.

Li, J.; Monroe, W.; and Jurafsky, D. 2017. Understanding Neural Networks through Representation Erasure. arXiv:1612.08220.

Miao, S.-Y.; Liang, C.-C.; and Su, K.-Y. 2021. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. Technical Report arXiv:2106.15772, arXiv.

Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models really able to Solve Simple Math Word Problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2080–2094.

Wallace, E.; Tuyls, J.; Wang, J.; Subramanian, S.; Gardner, M.; and Singh, S. 2019. AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models.

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep Neural Solver for Math Word Problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854. Copenhagen, Denmark: Association for Computational Linguistics.

Xie, Z., and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 5299–5305.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and LIM, E.-p. 2020. Graph-to-tree learning for solving math word problems. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* 3928–3937.

10 Appendix

Perturbation	**Original Question**	**Perturbed Question**	**Correct Equation**
Verbs Removed	Tommy had some balloons . His mom gave him number0 more balloons for his birthday . Then , Tommy had number1 balloons . How many balloons did Tommy have to start with ?	Tommy some balloons . His mom him number0 more balloons for his birthday . Then , Tommy number1 balloons . How many balloons Tommy to with ?	- number1 number0
Nouns Removed	The first minute of a telephone call costs number0 cents and each additional minute number1 cents . What is the cost of a number2 minute telephone call ?	The first of a number0 and each additional number1 . What is the of a number2 ?	+ number0 * number1 number2
Nouns and Verbs Removed	Virginia starts with number0 eggs . Amy takes number1 away . How many eggs does Virginia end with ?	with number0 . number1 away . How many with ?	- number0 number1
Prepositions and Verbs Removed	In March it rained number0 inches . It rained number1 inches less in April than in March . How much did it rain in April ?	March it number0 inches . It number1 inches less April March . How much it April ?	- number0 number1

Table 2: Examples of perturbed MWPs from the MaWPS dataset. In this dataset, the actual numbers are removed and replaced with number tokens (“number0”, “number1”, etc.) in order for the model to process them more easily.

Perturbation	MaWPS CV Accuracy	MaWPS Decrease in Accuracy	ASDiv-A CV Accuracy	ASDiv-A Decrease in Accuracy
original dataset	0.857	-	0.716	-
common adjectives removed	0.841	0.017	0.67	0.046
wh-adjectives removed	0.852	0.004	0.703	0.013
all adjectives removed	0.836	0.021	0.673	0.043
named entities removed	0.837	0.02	-	-
nouns removed	0.699	0.158	0.63	0.086
prepositions removed	0.832	0.025	0.683	0.033
verbs removed	0.806	0.051	0.613	0.103
nouns and verbs removed	0.553	0.304	0.519	0.197
prepositions and verbs removed	0.726	0.13	0.582	0.134
only nouns and number tokens remaining	0.232	0.625	0.217	0.499
only prepositions and number tokens remaining	0.125	0.732	0.193	0.523
only verbs and number tokens remaining	0.197	0.66	0.253	0.463
all words except number tokens removed	0.122	0.735	0.171	0.545

Table 3: Seq2seq model CV accuracy and decrease in CV accuracy on each perturbed dataset.

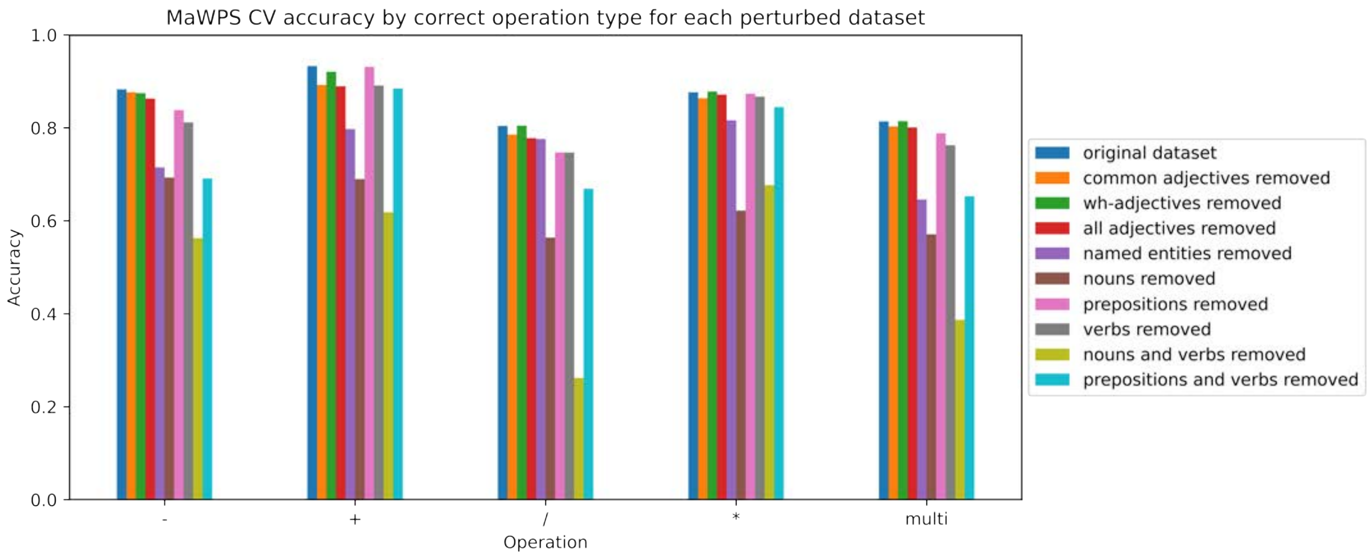


Figure 5: The RNN Seq2seq model's accuracy on each type of problem for the perturbed datasets with one or two parts of speech missing.

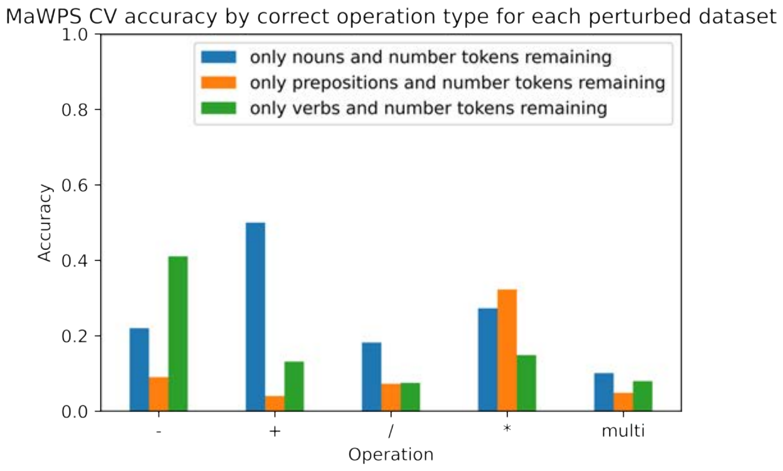


Figure 6: The RNN Seq2seq model's accuracy on each type of problem for the perturbed datasets with only one part of speech and number tokens remaining.

Number of Reductions	Prediction	Model Confidence	Removed Word	Question
0	Correct	0.927575		Bryan took a look at his books as well . If Bryan has number0 books in each of his number1 bookshelves , how many books does he have in total ?
1	Correct	0.999928	in	Bryan took a look at his books as well . If Bryan has number0 books each of his number1 bookshelves , how many books does he have total ?
2	Correct	0.999993	bookshelves	Bryan took a look at his books as well . If Bryan has number0 books each of his number1 , how many books does he have total ?
3	Correct	0.999994	has	Bryan took a look at his books as well . If Bryan number0 books each of his number1 , how many books does he have total ?
4	Correct	0.999995	bryan	took a look at his books as well . If number0 books each of his number1 , how many books does he have total ?
5	Correct	0.999992	how	took a look at his books as well . If number0 books each of his number1 , many books does he have total ?
6	Correct	0.999993	took	a look at his books as well . If number0 books each of his number1 , many books does he have total ?
7	Correct	0.999994	books	a look at his as well . If number0 each of his number1 , many does he have total ?
8	Correct	0.999989	.	a look at his as well If number0 each of his number1 , many does he have total ?
9	Correct	0.999992	if	a look at his as well number0 each of his number1 , many does he have total ?
10	Correct	0.999993	well	a look at his as number0 each of his number1 , many does he have total ?
11	Correct	0.999988	,	a look at his as number0 each of his number1 many does he have total ?
12	Correct	0.99999	as	a look at his number0 each of his number1 many does he have total ?
13	Correct	0.999988	at	a look his number0 each of his number1 many does he have total ?
14	Correct	0.999986	of	a look his number0 each his number1 many does he have total ?
15	Correct	0.999987	does	a look his number0 each his number1 many he have total ?
16	Correct	0.99999	total	a look his number0 each his number1 many he have ?
17	Correct	0.999992	he	a look his number0 each his number1 many have ?
18	Correct	0.999994	a	look his number0 each his number1 many have ?
19	Correct	0.999984	look	his number0 each his number1 many have ?
20	Correct	0.999799	?	his number0 each his number1 many have
21	Correct	0.990361	have	his number0 each his number1 many
22	Incorrect	0.616872	his	number0 each number1 many

Table 4: An example of the input reduction process.

Utilizing priming to identify optimal class ordering to alleviate the problems of catastrophic forgetting

Gabriel Mantione-Holmes
Lewis & Clark College
gabriel@lclark.edu

Justin Leo
University of Colorado
Colorado Springs
jleo@uccs.edu

Jugal Kalita
University of Colorado
Colorado Springs
jkalita@uccs.edu

Abstract

In order for artificial neural networks to begin accurately mimicking biological ones, they must be able to adapt to new exigencies without forgetting what they have learned from previous training. Lifelong learning approaches to artificial neural networks attempt to strive towards this goal, yet have not progressed far enough to be realistically deployed for NLP tasks. The proverbial roadblock of catastrophic forgetting still gate-keeps researchers from an adequate lifelong learning model. While efforts are being made to quell catastrophic forgetting, there is a lack of research that looks into the importance of class ordering when training on new classes for incremental learning. This is surprising as the ordering of "classes" that humans learn is heavily monitored and incredibly important. While heuristics to develop an ideal class order have been researched, this paper examines class ordering as it relates to priming as a scheme for incremental class learning. By examining the connections between various methods of priming found in humans and how those are mimicked yet remain unexplained in life-long learning, this paper provides a better understanding of the similarities between our biological systems and the synthetic systems while simultaneously improving current practices to combat catastrophic forgetting. Through the merging of psychological priming practices with class ordering, this paper was able to identify a generalizable method for class ordering in NLP incremental learning tasks that consistently outperforms random class ordering.

1 Introduction

Artificial neural networks have surpassed human abilities on a front of tasks. Human brains, unlike their synthetic counterparts, are hardly static in that they can, over their lifetime, learn new tasks while still retaining the ability to perform previously learned tasks. The same cannot be said about our current methods for isolated learning (Chen and Liu 2016). Isolated learning, while proven useful, has real world limitations in that it cannot use previously learned knowledge to facilitate learning new tasks while still retaining information. When an isolated learning model is retrained on new data, it typically suffers from *catastrophic forgetting*. Catastrophic forgetting, also known as *catastrophic interference* (McCloskey and Cohen 1989) is the process by which a model loses the ability to classify data on which it has previously been trained (Li, Qu, and Haffari 2021).

In order to perform well outside the lab, NLP models must be continually trained on batches of data from new classes while maintaining high accuracy in previously trained classification tasks. This paradigm is variously called Incremental learning, continual learning, sequential learning, or lifelong learning (Leo and Kalita 2022). Lifelong learning can benefit NLP research in ways that multi-class learning and isolated learning cannot. The major oversight shared by multi-class and isolated learning is that they assume data during training represent all the data and tasks that will be encountered in the real world. Lifelong learning, on the other hand, assumes that new tasks and data will naturally present themselves later. While lifelong learning models in computer vision have claimed a modicum of success, the mitigation of catastrophic forgetting among NLP models still has not met the metrics well (Greco et al. 2019).

Catastrophic forgetting is being combatted in many different ways. However, class ordering is very rarely examined as a means to alleviate the effects of catastrophic forgetting. Class ordering is the idea that the way class data are arranged and fed to an incremental classifier can affect the performance of the model being trained. Class ordering in relation to synthetic neural networks has been examined since the late 80s. Notions of ordering sensitivity have been introduced as a measure of the importance of the order in which a network is fed classes. While class ordering has been examined in class incremental learning (He, Wang, and Chen 2022; **?**), past work seems to only examine vision tasks.

To better understand class ordering within the NLP domain, this paper examines different methods of priming. Semantic, associative and repetition priming methods for NLP systems are inspired by the psychological practice of priming humans brains. While current methods of class ordering utilize information contained in confusion matrices (Masana, Twardowski, and Van de Weijer 2020), these require a model to first be trained on the classes to determine which classes the model most often mis-classifies.

The method that this paper proposes examines class data before training to determine an optimal ordering that limits the effects of catastrophic forgetting. This method should perform better outside the lab because as new classes are discovered, the class ordering can be modified to accommodate them. Examining semantic and associative relatedness between classes allows us to generate orders for classes that bank on artificial neural networks behaving analogously to the organic phenomenon of priming.

This paper aims to:

- Identify a generalizable class ordering method that outperforms random ordering for NLP incremental classification,
- Identify a class ordering method that can order data before model training occurs,
- Draw a connection between biological neural networks and artificial neural networks in the context of priming.

2 Related Work

This work examines the relationship between priming as it relates to humans and how priming can be applied to the NLP incremental classification paradigm. This relationship is examined in order to identify an optimal class ordering method.

2.1 Open-set classification

Most classification problems in machine learning are evaluated in the "closed-set" paradigm. This is the scenario in which the set of classes used in training is same as the set of classes used in testing. This represents the real world inaccurately and begs for the more realistic scenario of "open-set" classification. Open-set classification is the process of training a model on a set of classes that is less extensive than the set of tested classes (Scheirer et al. 2013). Open-set classification has been implemented using loss functions that increase the entropy of softmax scores to better handle background and unknown inputs (Dhamija, Günther, and Boult 2018). Others have replaced the softmax layer with the "Openmax" (Bendale and Boult 2016) layer that computes how far a piece of data is from known training data based on the penultimate layer's outputs (Prakhya, Venkataram, and Kalita 2017).

2.2 Class ordering

Class ordering during classification has been explored since the 1980s (Lee et al. 1988). However, the majority of efforts in class ordering have only examined vision (Yang and Li 2021). Efforts at ordering typically rely on a confusion matrix that results from a network already training on the entire dataset. To the knowledge of the authors, this is the first paper to examine class ordering for NLP tasks, as well as class ordering that can be implemented prior to any model training in the context of incremental class learning. As a final claim, we believe this paper is the first to examine the close relationship that biological and synthetic networks have regarding priming.

2.3 Priming

Priming is the psychological process by which one exposure to a certain stimulus has an influence on the reaction to the exposure of another stimulus (Bargh and Chartrand 2014). Psychologists have discovered three types of priming that the authors have identified as viable to be transferred to the incremental learning domain. *Semantic priming* (Shelton and Martin 1992) refers to priming in which the initial stimulus has an important semantic relationship to the reacted stimulus, which influences the reaction to a greater degree than some other stimulus that had a weaker semantic relationship. *Associative priming* (Ferrand and New 2004) refers to priming in which the initial stimulus is in close proximity to the reacted stimulus (and hence associated), which influences the reaction to a greater degree than some other stimulus that appears further away in proximity. The final priming method is *repetition priming* (Forster and Davis 1984). This is priming in which the initial stimulus affects future reactions to itself. As we are examining incremental learning in NLP, these three methods stood out to us. Being able to compute words that have semantic and associative relationships to a given word enables us to see if artificial neural networks are affected similarly to biological neural networks. Our model relies on rehearsal strategies (Luo et al. 2020) where past data are used for retraining when a new class is discovered. This incremental learning method has deep roots in repetition priming as both are a form of replay of data.

3 Problem Statement

While lifelong learning has had some success in computer vision, the same cannot be said for NLP. Lifelong learning models still suffer from catastrophic forgetting when faced with NLP tasks. Class order has been examined for lifelong learning in the context of computer vision, but has been completely untouched in NLP. This paper identifies class ordering methods that outperform random class ordering, which increases accuracy of the model, and hence hampers catastrophic forgetting.

4 Approach

The goal of this paper is to identify a method for class ordering that will outperform random class ordering for NLP classification tasks. Four different methods were tested to identify if there was a method that outperformed random order. We also tested whether interleaved or block learning (He, Wang, and Chen 2022) performs better in conjunction with the four different methods. The model used to test these class ordering methods was the Classification Confidence Threshold (CCT) model (Leo and Kalita 2021). The CCT model identifies new classes using open-set classification and a confidence threshold. The presence of data from previously unseen classes is identified through spectral clustering of unknown pieces of data. Examples of previously unseen classes are identified through a scheme where the classifier is "primed" to be on the lookout for data from unseen classes. A confidence threshold is used to determine whether or not the class to which the data example belongs is truly unknown. When the classifier is retrained on the newly identified class data, the model adds new nodes at the output layer to catch new class data.

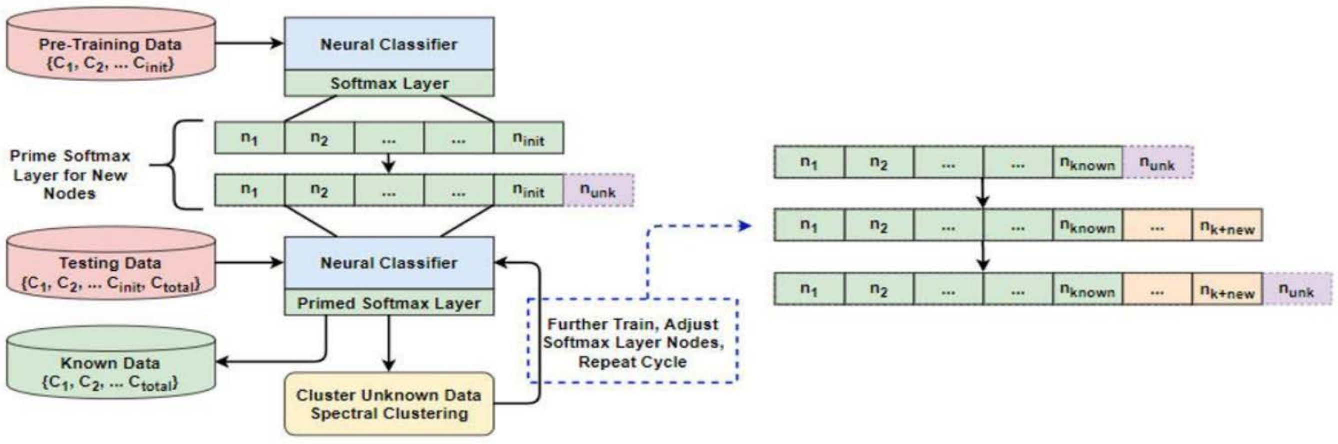

Pre-Training Data {C1, C2, ... Cinit}
Neural Classifier
Softmax Layer
Prime Softmax Layer for New Nodes
n1 n2 ... ... ninit
n1 n2 ... ... ninit nunk
Testing Data {C1, C2, ... Cinit, Ctotal}
Neural Classifier
Primed Softmax Layer
Known Data {C1, C2, ... Ctotal}
Cluster Unknown Data Spectral Clustering
Further Train, Adjust Softmax Layer Nodes, Repeat Cycle
n1 n2 ... ... nknown nunk
n1 n2 ... ... nknown ... nk+new
n1 n2 ... ... nknown ... nk+new nunk

Figure 1: A representation of the CCT method that adds a priming node at the classifiers softmax layer (Leo and Kalita 2021).

4.1 Associative Priming

The first heuristic we examined was based on associative relationships between classes. This heuristic was pursued due the effects of associative priming on human learning. To find the associative relationships between classes, an exemplar of the class is first identified by computing the average document for a class. The average document is computed by taking the average of all word embeddings of all documents in a class. While this is a relatively naive method of making an exemplar for a text class, it is computationally cheap and can be computed without the training of anything but a word2vec model (Mikolov et al. 2013) for the corpus. Along with using word2vec, doc2vec was also used to determine an exemplar by computing the average document as the average of doc2vec representations (Le and Mikolov 2014). However, since most of the documents are short, our approach is practical and resonably sound.

After an exemplar is computed for each class, the similarity is computed by using cosine similarity. The similarity/dissimilarity between pairs of classes is then stored in an adjacency matrix. The distance between class averages is also computed using the Euclidean distance between them, as an alternative. While cosine similarity tells us how similar classes are to one another Euclidean distance informs us of how different classes are and where their is little difference similarity is assumed.

Once an adjacency matrix is computed, an order is constructed based on *interleaved learning*, the process of learning data classes that differ significantly, or *block learning*, the process of learning data classes that are very similar (He, Wang, and Chen 2022). For interleaved learning, the order is constructed by taking the pair of classes with the greatest distance or least similarity as the first two classes and adding on a class to the order that is furthest/ least similar to the last class added. This results in an order of classes that vary the most from each other on an associative level. For block learning, a similar method is used, but instead of using classes have the greatest distance/lowest similarity, we take classes that have the least distance or most similarity.

While this method can be made more complex by identifying more robust methods of computing association between classes, this current naive approach offers a starting point for future research in this area.

4.2 Semantic Priming

The second heuristic we examined was based on semantic relationships between classes. This heuristic was pursued also due to the success of semantic priming within the human learning system. To find the semantic relationships between classes, the set of top 10 most important words per class was computed using tf-idf. For each set of words, the semantic similarity was computed. This was done by obtaining the WordNet synset of each word and computing the Wu-Palmer similarity (Wu and Palmer 1994) between them. Once the similarity between each word of the two sets is computed, the average similarity between the words of the two sets is computed and used as the semantic similarity between the two classes. This is done between each class which results in an adjacency matrix.

Once an adjacency matrix is computed, the order is constructed again based on either block learning where the order will be comprised of semantically similar classes or interleaved learning where the order will be comprised of semantically dissimilar classes.

4.3 Repetition Priming

For semantic and associative priming, it is then examined whether repetition priming of semantic and associative classes has any benefits. Because the incremental learning model we are using relies on rehearsal, the incremental learner is already using repetition priming by using the same order. To determine whether repetition affects this process, the model was trained on the full order as well as using the order for the first five classes and then using a random order for the rest. This acts as an ablation study on the effectiveness of repetition priming when it comes to class ordering.

4.4 Data

Two text datasets were used to determine if a method of class ordering could be generally applied across incremental learning NLP tasks. The Reddit Mental Health Dataset (RMHD) consists of posts from 27 subreddits (Low et al. 2020). The CCAT-50 dataset consists of 50 writing pieces from 50 authors (Houvardas and Stamatatos 2006). These two datasets differ greatly in that one consists of internet posts that are filled with slang and vernacular text while CCAT-50 consists of formal writing.

5 Results

In Figs 2 and 3, we present averaged experimental data using the RMHD and CCAT-50 datasets, respectively. The figures show the accuracy of the model over 15 iterations using associative priming methods with word2vec. The model starts with training on five classes and with each iteration adds one new class. In both Figs 2 and 3, we see how associative interleaved priming performs much better than associative block priming. This mimics what psychologists have found to be true about humans (Rohrer, Dedrick, and Stershic 2015; Pan 2015). Not only is it interesting that the CCT model mimics the same priming effect as humans, but ordering based on associative priming outperforms random ordering by about 15% for RMHD and 20% for CCAT-50. The interleaving priming outperforms block priming by 20% for RHMD and 10% for CCAT-50. Across both datasets associative interleaved priming outperforms random.

In Figs 4 and 5, we present average experimental data for semantic priming for our datasets. While semantic block priming outperforms interleaved priming for RMHD, this trend is not supported by CCAT-50. The discrepancies arising in the semantic priming results can be explained by the shortcomings of Wordnet. While Wordnet is a fantastic tool, if a word is not in Wordnet's vocabulary, it will not be able to create a synset for it. When examining the most important words for both datasets many sets of top words included slang, proper nouns, and acronyms all of which were overlooked by Wordnet. Until semantic similarity can be more easily computed between words that are currently overlooked, this research concludes that semantic priming is a dead end.

In Figs 6 and 7 we present average experimental data for associative priming using doc2vec rather than word2vec. Surprisingly the use of doc2vec does not appear to produce conclusive results across both datasets. While there are methods that outperform random ordering within doc2vec associative priming for both datasets, they do not behave similarly and therefore the argument for one of them working well throughout the NLP domain is invalidated.

.

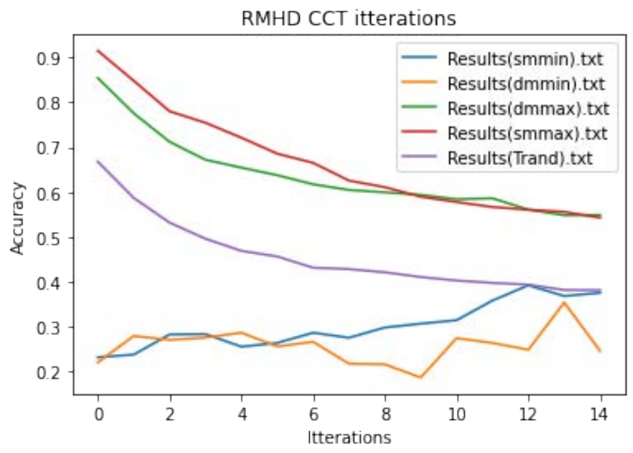


Figure 2: Averaged results from associative priming for the RMHD dataset using word2vec

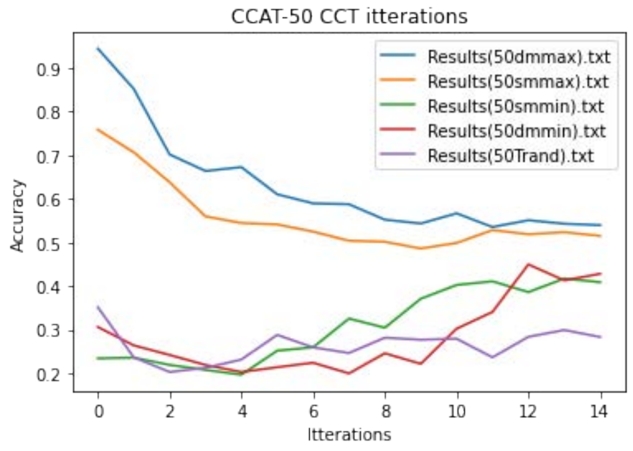


Figure 3: Averaged results from associative priming for the CCAT-50 dataset using word2vec

6 Conclusion

Class ordering does matter for NLP incremental learning tasks. In our process of examining class ordering, we have found a method that improves performance over random class ordering. In addition, we have found another method that underperforms random ordering. Class ordering appears to perform best across both datasets when associated-interleaved-repetition priming is utilized. Furthermore, since the same is true for humans, this discovery has revealed a similarity between biological and synthetic networks. Reproduction with other datasets will help solidify our claim that this method is truly generalizable. This work does point toward the need to look further into class ordering in the context of incremental NLP classification. Random order is consistently outperformed with one of the methods we have used. We conclude that associative-interleaved-repetition priming should be further examined across various NLP incremental classification problems.

7 Acknowledgements

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations ex-

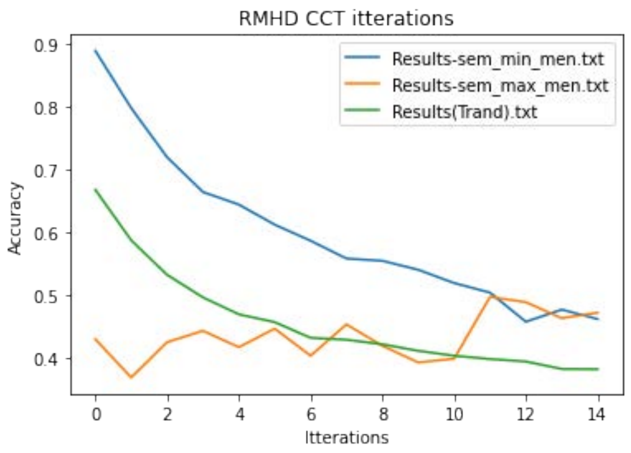


Figure 4: Averaged results from semantic priming for the RMHD dataset

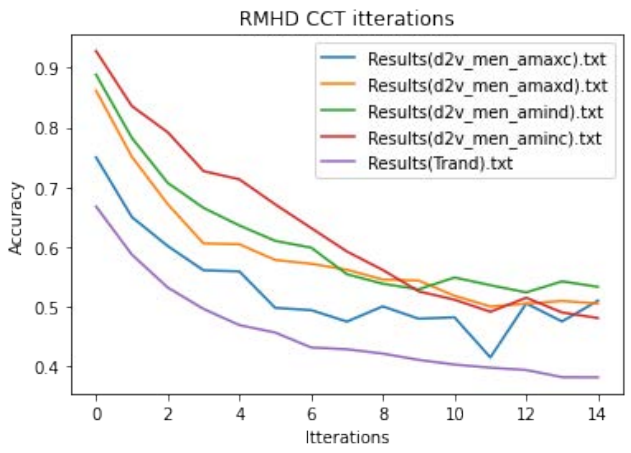


Figure 6: Averaged results from associative priming for the RMHD dataset using doc2vec

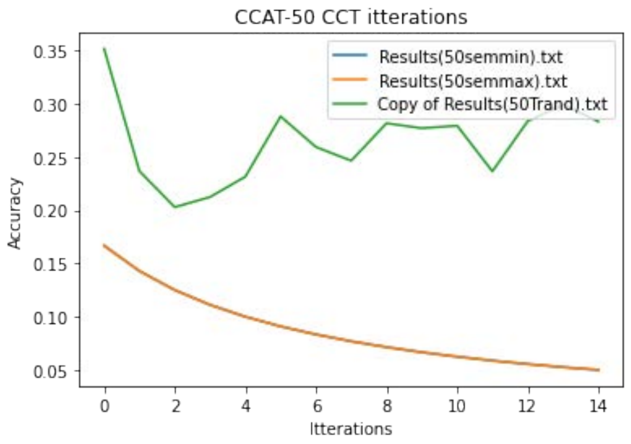


Figure 5: Averaged results from semantic priming for the CCAT-50 dataset

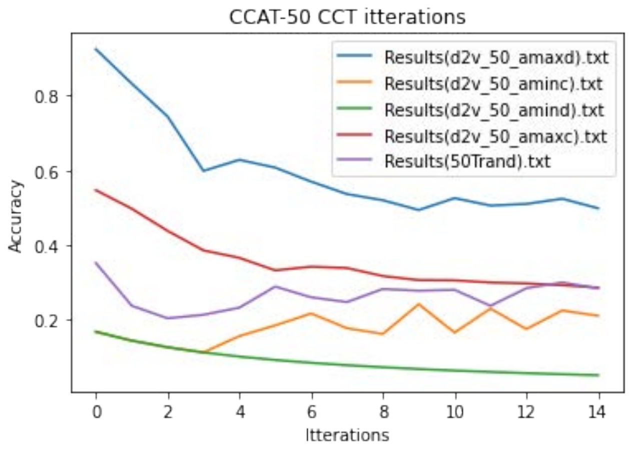


Figure 7: Averaged results from associative priming for the CCAT-50 dataset using doc2vec

pressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Bargh, J. A., and Chartrand, T. L. 2014. The mind in the middle: A practical guide to priming and automaticity research.

Bendale, A., and Boult, T. E. 2016. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, Z., and Liu, B. 2016. Lifelong machine learning for natural language processing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*. Austin, Texas: Association for Computational Linguistics.

Dhamija, A. R.; Günther, M.; and Boult, T. 2018. Reducing network agnostophobia. *Advances in Neural Information Processing Systems* 31.

Ferrand, L., and New, B. 2004. Semantic and associative. *Mental lexicon: Some words to talk about words* 25.

Forster, K. I., and Davis, C. 1984. Repetition priming and frequency attenuation in lexical access. *Journal of experimental psychology: Learning, Memory, and Cognition* 10(4):680.

Greco, C.; Plank, B.; Fernández, R.; and Bernardi, R. 2019. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3601–3605. Florence, Italy: Association for Computational Linguistics.

He, C.; Wang, R.; and Chen, X. 2022. Rethinking class orders and transferability in class incremental learning. *Pattern Recognition Letters*.

Houvardas, J., and Stamatatos, E. 2006. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*, 77–86. Springer.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, 1188–1196. PMLR.

Lee, E. S.; MacGregor, J. N.; Bavelas, A.; Mirlin, L.; Lam, N.; and Morrison, I. 1988. The effects of error transformations on classification performance. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 14(1):66.

Leo, J., and Kalita, J. 2021. Incremental Deep Neural Network Learning Using Classification Confidence Thresholding. *IEEE Transactions on Neural Networks and Learning Systems* 1–11.

Leo, J., and Kalita, J. 2022. Survey of Continuous Deep Learning Architectures for Incremental Learning. unpublished manuscript.

Li, Z.; Qu, L.; and Haffari, G. 2021. Total Recall: a Customized Continual Learning Method for Neural Semantic Parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3816–3831. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Low, D. M.; Rumker, L.; Talkar, T.; Torous, J.; Cecchi, G.; and Ghosh, S. S. 2020. Natural Language Processing Reveals Vulnerable Mental Health Support Groups and Heightened Health Anxiety on Reddit During COVID-19: Observational Study. *J Med Internet Res* 22(10):e22635.

Luo, Y.; Yin, L.; Bai, W.; and Mao, K. 2020. An Appraisal of Incremental Learning Methods. *Entropy* 22(11):1190.

Masana, M.; Twardowski, B.; and Van de Weijer, J. 2020. On class orderings for incremental learning. *arXiv preprint arXiv:2007.02145*.

McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*. Academic Press. 109–165.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pan, S. C. 2015. The interleaving effect: mixing it up boosts learning. *Scientific American* 313(2).

Prakhya, S.; Venkataram, V.; and Kalita, J. 2017. Open-set deep learning for text classification. *Machine Learning in Computer Vision and Natural Language Processing; ACM: New York, NY, USA* 1–6.

Rohrer, D.; Dedrick, R. F.; and Stershic, S. 2015. Interleaved practice improves mathematics learning. *Journal of Educational Psychology* 107(3):900.

Scheirer, W. J.; de Rezende Rocha, A.; Sapkota, A.; and Boult, T. E. 2013. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(7):1757–1772.

Shelton, J. R., and Martin, R. C. 1992. How semantic is automatic semantic priming? *Journal of Experimental Psychology: Learning, memory, and cognition* 18(6):1191.

Wu, Z., and Palmer, M. 1994. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*.

Yang, Z., and Li, H. 2021. Task ordering matters for incremental learning. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–6.

CAMeMBERT: Cascading Assistant-Mediated Multilingual BERT

Dan DeGenaro
University of Massachusetts, Amherst
Department of Linguistics
650 North Pleasant St.
Amherst, MA 01003
ddegenaro@umass.edu

Jugal Kalita
University of Colorado, Colorado Springs
Department of Computer Science
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
jkalita@uccs.edu

Abstract

Massive language models having hundreds of millions, and even billions, of parameters have performed extremely well on a variety of natural language processing (NLP) tasks. Their widespread use and adoption, however, is hindered by the lack of availability and portability of sufficiently large computational resources. This paper proposes a knowledge distillation (KD) technique building on the work of LightMBERT, a student model of multilingual BERT (mBERT). By repeatedly distilling mBERT through increasingly compressed top-layer distilled teaching assistant networks, CAMeMBERT aims to improve upon the time and space complexities of mBERT while keeping loss of accuracy beneath an acceptable threshold. At present, CAMeMBERT has an average accuracy of around 60.1%, which is subject to change after making improvements to the hyperparameters used in fine-tuning.

1 Introduction

Massive multilingual language models such as multilingual BERT (mBERT) have excelled at tasks such as machine translation, question answering, and structured predictions (Hu et al. 2020). However, they are generally too computationally expensive to be used on personal devices. A solution to this problem is knowledge distillation (KD). KD was first suggested by Buciluă, Caruana, and Niculescu-Mizil (2006), but was first popularized among machine learning researchers by Hinton, Vinyals, and Dean (2015). KD refers to the idea of “distilling” a larger model into a much smaller one, often called “teacher” and “student” networks, respectively. Many successful techniques have been employed in order to reduce the computational needs of these impressive neural networks, while maintaining relatively small losses in accuracy. A simple approach to KD may be, for instance, using a loss function that minimizes the difference between the logits of the teacher and student models.

- This paper aims to improve upon the results of (Jiao et al. 2021) by applying a similar distillation technique, combined with use of teacher assistant networks as described by (Mirzadeh et al. 2020).
- In so doing, this paper also proposes use of adjacent layer averaging as a teacher-to-student layer mapping during the distillation process.

What follows is a brief review of related work (Section 2), a fuller description of both the problem at hand (Section 3) and our approach to solving it (Section 4), and an evaluation of our model’s performance on the XNLI metric (Conneau et al. 2018), a common benchmark for multilingual language models (Section 5).

2 Related Work

2.1 Initializing from Teacher Networks

Rather than directly training the student model on the training data using tasks like masked language modeling, it has been shown to be more effective to train the student model to just mimic the teacher (Gou et al. 2021). This has been done in several ways. One of the first methods employed, DistilBERT (Sanh 2020), involved training a new, smaller network (half as many layers) to mimic the original model. DistilBERT performed extremely well on various benchmarks, and was initialized with one out of every two layers of its teacher. This approach led to a slightly different student initialization method known as “top-layer distillation” (Jiao et al. 2021), in which the student network is initialized with the lower layers of the teacher network, and trained in similar fashion to mimic the teacher. In this paper, the student network, LightMBERT, was initialized with the lower six encoder layers of mBERT, and then trained to mimic the teacher mBERT.

2.2 Teacher Assistant Networks

In computer vision (as well as other fields employing neural networks), an increasingly popular framework for KD makes use of “teaching assistant” (TA) networks that try to bridge the large gap in capabilities between student and teacher networks (Mirzadeh et al. 2020). TAKD has been shown to improve retention of information by student networks. Indeed, it has been shown by Mirzadeh et al. that a distillation path having the maximal number of TAs produces optimal results (removing only one or two layers at a time).

2.3 Other Techniques

Other work has focused on reducing the amount of memory needed by storing weight matrices into tensor products (Tahaei et al. 2021) as well as mixing traditional learning with distillation, weighted by a monotonically decreasing temperature hyperparameter (Jafari et al. 2021). TinyBERT (Jiao et al. 2020) used a two stage training method which factored in loss not only in the pre-trained model, but in the fine-tuning layer as well.

3 Problem Statement

Large language models are simply too cumbersome and slow on inference for most potential users. While KD is being actively researched as a solution to this issue, multilingual models, in particular, are largely ignored in favor of continually improving English-language models, such as BERT (Devlin et al. 2019), and other single-language models. If sufficiently reduced in size without much loss of accuracy, a smaller, faster, distilled network is far better suited to use on edge devices with limited resources than are larger models. We build a six-layer BERT-style network following Jiao et al. (2021) that aims to improve upon their results, namely by employing iterated assistant network distillation as described by Mirzadeh et al. (2020).

4 Approach

4.1 High-level Details

Our approach combines several techniques that have been successfully employed independently, namely top-layer distillation (Jiao et al. 2021) and iterated assistant networks (Mirzadeh et al. 2020). CAMeMBERT was constructed via iterated top-layer distilled assistant networks. Starting from the 12-layer mBERT network, an assistant network is initialized with the lowest 11 layers of mBERT, including both weights and architecture (cutting out the topmost layer). This assistant network is then trained to mimic the teacher (mBERT). Then, a second assistant network is constructed using the first, albeit again with the top encoder layer removed. The second network is initialized with the lowest ten layers of the first assistant, both weights and architecture. This process is iterated until a network of six hidden layers is left, which is the CAMeMBERT model. This model is the same size as LightMBERT (Jiao et al. 2021). See Figure 1 for a diagram of this process.

4.2 Low-level Details

Pretraining Data The training corpus consists of a single text file of about 43 GB. Parts of the file were not used, however. It contains lines of text scraped from the largest 104 languages on Wikipedia[1] (Wikimedia Foundation 2022), as described by Devlin et al. (2019). The languages were sampled by a probability distribution P' defined as

$$P'(\text{lang}_j) = \frac{P(\text{lang}_j)^S}{\sum_k P(\text{lang}_k)^S} \tag{1}$$

where $P(\text{lang}_j)$ denotes the probability of language j according to the proportional space it occupies on disk, i.e.

$$P(\text{lang}_j) = \frac{\text{size}(\text{lang}_j)}{\sum_k \text{size}(\text{lang}_k)} \tag{2}$$

S was chosen such that $P'(\text{English}) = 100P'(\text{Icelandic})$, as described by the BERT team[2]. The texts used to produce the corpus were cleaned to contain only the plain text of an article, including headings, and blank lines were removed. The articles from all languages were then concatenated into one large text file, whose lines were rearranged in random order using the GNU/Linux `shuf` utility. Each line was considered one training example (so a batch size of 256 would involve accumulating loss over 256 lines, one at a time, before performing an optimizer step).

Table 1: Language abbreviations and representation.

Language	Abbreviation	% of corpus
English	en	12.0
Spanish	es	3.49
Chinese	zh	2.08
German	de	6.06
Arabic	ar	2.33
Urdu	ur	0.365

Pretraining Method Each network was trained for 66,666 steps (batches), for a total of about 400,000 steps across all six networks. Each pretraining distillation process used a distinct section of the corpus, unseen by preceding or following networks. The mBERT pretrained tokenizer from HuggingFace was used. This tokenizer is cased, and at no point was lowercasing was performed on the text. The vocabulary size for this tokenizer is 119,547. Maximum length padding was employed, along with truncation, with a maximum input sequence length of 128. All networks were kept in training mode (i.e. dropout enabled) throughout the process, and embedding layers were always frozen. Each network was optimized via an Adam optimizer defined by the following hyperparameters: batch size 256, peak learning rate 1×10^{-7}, linear warmup over the first 6,666 steps, linear decay, $\beta_{1,2} = 0.9, 0.999$, no weight decay, and $\epsilon = 1\times10^{-9}$. It was also noted that loss decreased more sharply when training the 11-layer network if linear warmup was applied throughout the training process, so this was done. These choices are summarized in Table 1.

The total loss for a batch is defined as follows, following Jiao et al. (2021):

$$\mathcal{L} = \frac{1}{n}\left(\sum_{j=1}^{n}\mathcal{L}_j^A + \sum_{k=1}^{n+1}\mathcal{L}_k^H\right) \tag{3}$$

where the student network has n attention layers and $n+1$ hidden outputs (this implies the teacher has $n+1$ attention layers and $n+2$ hidden outputs). $\mathcal{L}_k^H$ is the loss of one layer's

[1]The text was obtained via HuggingFace's Datasets library, except for the languages Cebuano and Spanish, which were obtained via the WikiExtractor tool (all from the 2022-03-01 dump).

[2]Consider looking at this markdown file in the GitHub repo for BERT, which gives more details: https://github.com/google-research/bert/blob/master/multilingual.md

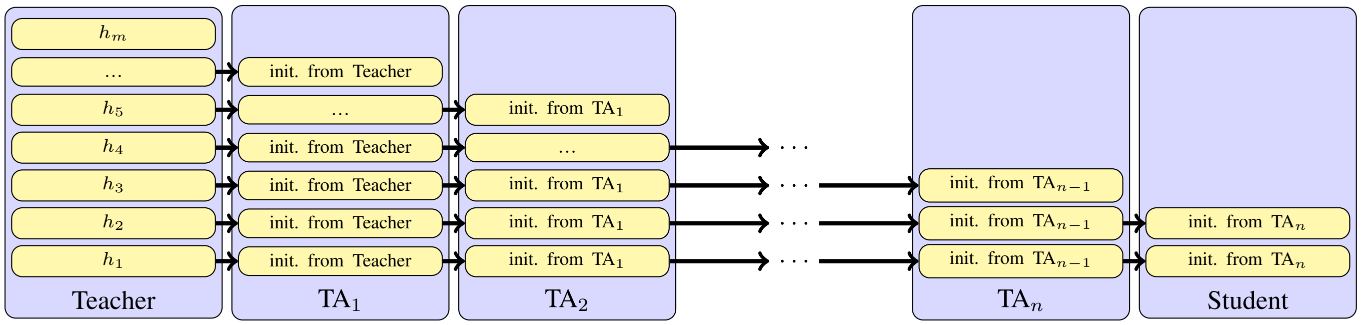


Figure 1: Repeated top-layer distillation via teaching assistant networks (embeddings not shown). The number of hidden layers in network j is one fewer than in network $j-1$.

hidden outputs, and $\mathcal{L}_j^A$ is the loss of one layer's attentions. These are defined thus:

$$\mathcal{L}_j^A = \frac{1}{12}\sum_{\ell=1}^{12} \text{MSE}\left(\frac{1}{2}\left(A_T^{j\ell} + A_T^{j+1,\ell}\right), A_S^{j\ell}\right) \quad (4)$$

$$\mathcal{L}_k^H = \text{MSE}\left(\frac{1}{2}\left(H_T^k + H_T^{k+1}\right), H_S^k\right) \quad (5)$$

where $_{T,S}$ denote teacher and student, respectively, and ℓ ranges over the attention heads, of which there are 12 in the case of a $\text{BERT}_{\text{BASE}}$-style network. $\mathcal{L}_j^A$ is therefore the average loss of attention over the 12 attention heads at the j-th layer.

It should be noted that $\mathcal{L}_j^A$ and $\mathcal{L}_k^H$ are defined differently from Jiao et al. (2021) out of necessity; in that paper, a "layer mapping" can be performed from 12 layers to six layers directly, but in this paper, since one layer is removed at a time, the mapping must map 12 to 11, 11 to ten, and so on. Thus, the mapping is defined (as is evident from the definition of the loss at one layer) to be the average of layers j and $j+1$ of the teacher to the j-th layer of the student, a strategy we call "adjacent layer averaging." To be clear, the average of layers six and seven of a teacher would be used to train the sixth layer of its student, for instance. See Figure 2 for a diagram of this process.

It is also important to define MSE clearly. MSE refers to mean squared error, which is defined generically as:

$$\text{MSE}(X, Y) = \frac{1}{\text{numel}(X)} \sum_{j,k,\ldots} (X_{jk\ldots} - Y_{jk\ldots})^2 \quad (6)$$

where $\text{shape}(X) = \text{shape}(Y)$, both can be indexed by indices $j, k, \ldots$ and $\text{numel}(X)$ (which equals $\text{numel}(Y)$) denotes the number of elements in X (product of ranges of all indices).

Fine-tuning Method The networks were fine-tuned on the English XNLI dataset's "train" split obtained via HuggingFace's Datasets library. Fine-tuning was conducting by an Adam optimizer as follows: 3 epochs, batch size 32, learning rate 2×10^{-5}, $\beta_{1,2} = 0.9, 0.999$, no weight decay, and $\epsilon = 2 \times 10^{-7}$. Padding and truncation were applied with a maximum sequence length of 128. Our network's weights were loaded into a HuggingFace BertForSequenceClassification architecture, with weights of the fine-tuning architecture being randomly initialized. The number of output classes was set to be 3, as XNLI data are labeled as either 'entailment' (0), 'neutral' (1), or 'contradiction' (2). Cross entropy loss was employed as the fine-tuning loss[3].

Table 2: Hyperparameters. Note that the abbreviation 'LR' refers to learning rate.

Hyperparameter	Pretraining	Fine-tuning
optimizer	Adam	Adam
epochs	1	3
steps	66,666	12,271
batch size	256	32
peak LR	1×10^{-7}	2×10^{-5}
LR warmup	linear, first 6,666 steps*	none
LR decay	linear, to the end	none
β_1	0.9	0.9
β_2	0.999	0.999
weight decay	0	0
ϵ	1×10^{-9}	2×10^{-7}
padding	True	True
max seq. length	128	128
embeddings frozen	True	True
vocab size	119,547	119,547

*The first TA (11 layer network) had linear warmup over all 66,666 steps.

4.3 Evaluation Metric

The networks were evaluated on the zero-shot cross-lingual transfer task using HuggingFace's "test" split of the XNLI dataset. It was evaluated on the six languages present in Table 2, and the mean accuracy (AVG) displayed in Table 3 is the mean accuracy across those six languages.

[3]See documentation here: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

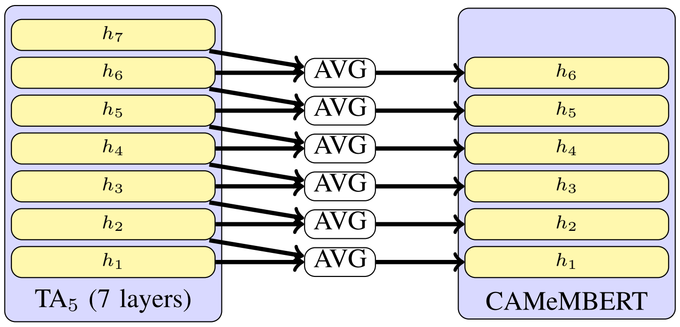


Figure 2: Adjacent layer averaging between a teacher network and its student. The number of hidden layers in network j is one fewer than in network $j-1$.

5 Results

At present, the networks do not perform as well as LightMBERT, but close to it. Improving these results will be a matter of adjusting the mapping encoded into the loss function, which currently averages two layers of a teacher to train one layer of a student, adjusting learning rates throughout the distillation process, and adjusting the number of warmup and decay steps, primarily in the fine-tuning process.

Table 3: Results on XNLI Task by Language and Number of Layers

	en	es	zh	de	ar	ur	AVG
11	81.0	73.3	69.3	70.1	64.6	57.1	69.2
10	79.2	71.3	66.1	67.0	61.8	56.4	67.0
9	79.9	69.6	65.9	65.5	60.9	56.4	66.4
8	77.8	67.8	64.9	64.2	58.9	54.5	64.7
7	77.7	65.7	64.6	62.3	57.1	52.3	63.3
6	76.8	62.1	60.7	60.1	51.5	49.6	60.1
LMB*	81.5	74.7	69.3	72.2	65.0	59.3	70.3

*LMB is LightMBERT's results (Jiao et al. 2021).

6 Conclusion

By applying repeated teaching assistant-mediated top-layer distillations to a large language model, this work stands to produce a fast, memory- and storage-efficient neural network that can mimic the abilities of mBERT. This work builds on that of LightMBERT, which was created via top-layer distillation, as well as that of the TAKD framework for more effective knowledge distillation.

7 Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Thanks to Abigail Swenor (Notre Dame), Aaron Serianni (Princeton), and Dr. Terrance Boult (UCCS) for their help with the practical side of this project.

References

Buciluǎ, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, 535–541. New York, NY, USA: Association for Computing Machinery.

Conneau, A.; Lample, G.; Rinott, R.; Williams, A.; Bowman, S. R.; Schwenk, H.; and Stoyanov, V. 2018. XNLI: Evaluating Cross-lingual Sentence Representations. Number: arXiv:1809.05053 arXiv:1809.05053 [cs] version: 1.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge Distillation: A Survey. *International Journal of Computer Vision* 129(6):1789–1819.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. Technical Report arXiv:1503.02531, arXiv. arXiv:1503.02531 [cs, stat] type: article.

Hu, J.; Ruder, S.; Siddhant, A.; Neubig, G.; Firat, O.; and Johnson, M. 2020. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, 4411–4421. PMLR. ISSN: 2640-3498.

Jafari, A.; Rezagholizadeh, M.; Sharma, P.; and Ghodsi, A. 2021. Annealing Knowledge Distillation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2493–2504. Online: Association for Computational Linguistics.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2020. TinyBERT: Distilling BERT for

Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4163–4174. Online: Association for Computational Linguistics.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2021. LightMBERT: A Simple Yet Effective Method for Multilingual BERT Distillation. Technical Report arXiv:2103.06418, arXiv. arXiv:2103.06418 [cs] type: article.

Mirzadeh, S. I.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; and Ghasemzadeh, H. 2020. Improved Knowledge Distillation via Teacher Assistant. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(04):5191–5198.

Sanh, V. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *EMC^2* 5:5.

Tahaei, M. S.; Charlaix, E.; Nia, V. P.; Ghodsi, A.; and Rezagholizadeh, M. 2021. KroneckerBERT: Learning Kronecker Decomposition for Pre-trained Language Models via Knowledge Distillation. Technical Report arXiv:2109.06243, arXiv. arXiv:2109.06243 [cs] type: article.

Wikimedia Foundation. 2022. Wikimedia downloads.

Training-free Neural Architecture Search for RNN and Transformer Architectures

Aaron Serianni,[1] **Jugal Kalita**[2]

[1]Princeton University,
[2]University of Colorado Colorado Springs
serianni@princeton.edu, jkalita@uccs.edu

Abstract

Neural architecture search (NAS) has allowed for the automation of creating new and effective neural network architectures, offering an alternative to the laborious process of manually designing complex architectures. However, traditional NAS algorithms are slow and require immense amounts of computing power. Recent research has investigated training-free NAS metrics for image classification architectures, drastically speeding up search algorithms. In this paper, we investigate for the first time training-free NAS metrics for recurrent neural network (RNN) and BERT-based transformer architectures, targeted towards language modeling tasks. First, we develop a new training-free metric, named hidden covariance, that predicts the trained performance of an RNN architecture, and outperforms existing training-free metrics. We experimentally evaluate the effectiveness of the hidden covariance metric on the NAS-Bench-NLP benchmark. Second, we find that the current search space paradigm for BERT-based models is not optimized for training-free neural architecture search. Instead, a simple qualitative analysis can effectively shrink the search space to the best performing models. This conclusion is based on our investigation of existing training-free metrics and new metrics developed from recent transformer pruning literature, evaluated on our own benchmark of trained BERT models. Ultimately, our analysis shows that the architecture search space and the training-free metric must be developed together in order to achieve effective results.

1 Introduction

Recurrent neural networks (RNNs) and BERT-based models with self-attention have been extraordinary successful in achieving state-of-the-art results on a wide variety of language modeling-based natural language processing (NLP) tasks, including question answering, sentence classification, tagging, and natural language inferencing (Brown et al. 2020; Palangi et al. 2016; Raffel et al. 2020; Sundermeyer, Schlüter, and Ney 2012; Yu et al. 2019). However, the manual development of new neural network architectures has become increasingly difficulty as models become larger and more complicated. Neural architecture search (NAS) algorithms aim to procedurally design and evaluate new, efficient, and effective architectures within a predesignated search space (Zoph and Le 2017). NAS algorithms have been extensively used for developing new convolutional neural network (CNN) architectures for image classification, with many surpassing manually-designed architectures and achieving SOTA results on many classification benchmarks (Tan and Le 2019; Real et al. 2019).

While NAS algorithms and methods have been successful in developing novel and effective architectures, there are two main problems that current algorithms face. The search space for various architectures is immense, and the amount of time and computational power to run NAS algorithms is prohibitively expensive (Mehta et al. 2022). Because traditional NAS algorithms require the evaluation of candidate architectures in order to gauge performance, each candidate architecture needs to be trained fully, taking hours or days to complete. Thus, past attempts at NAS have been critiqued for being computationally resource-intensive, consuming immense amounts of electricity, and producing large amounts of carbon emissions (Strubell, Ganesh, and McCallum 2019). These problems are particularly true for transformers and RNNs, as they have more parameters and take longer to train when compared to other types of neural networks (So, Le, and Liang 2019; Zhou et al. 2022).

Recently, there has been research into training-free NAS metrics and algorithms, which offer significant performance increases over traditional NAS algorithms (Abdelfattah et al. 2020; Mellor et al. 2021a; Zhou et al. 2022). These metrics aim to partially predict an architecture's trained accuracy from its initial untrained state, given a subset of inputs. However, prior research has focused on developing training-free NAS metrics for CNNs and Vision Transformers with image classification tasks. In this work, we apply existing and create our own training-free metrics for RNNs and BERT-based transformers with language modeling tasks. Our main contributions are:

- We develop a new training-free metric for RNN architectures, hidden covariance, which significantly outperforms existing metrics on NAS-Bench-NLP.
- We develop a NAS benchmark for BERT-based models utilizing the FlexiBERT search space and ELECTRA pretraining scheme.
- We evaluate existing training-free metrics on our NAS benchmark for BERT-based models, and propose a series of new metrics adapted from attention head pruning.
- Finally, we discuss current limitations with training-free NAS for transformers due to the structure of transformer

search spaces, and propose an alternative paradigm for speeding up NAS algorithms based on scaling laws of transformer hyperparameters.

2 Related Work

Since the development and adoption of neural architecture search, there has been research into identifying well-performing architectures without the costly task of training candidate architectures.

2.1 NAS Performance Predictors

Prior attempts at predicting a network architecture's accuracy focused on training a separate performance predictor. Deng, Yan, and Lin (2017) and Istrate et al. (2019) developed methods called Peephole and Tapas, respectively, to embed the layers in an untrained CNN architecture into vector representations of fixed dimension. Then, both methods trained LSTM networks on these vector representations to predict the trained architecture's accuracy. Both methods achieved strong linear correlations between the LSTMs' predicted accuracy and the actual trained accuracy of the CNN architectures. In addition, the LSTM predictors can quickly evalutate large amounts of CNN architectures. The primary limitation of these methods is that the LSTM predictors require large amounts of trained CNN architectures in order to accurately train the predictors, thus not achieving the goal of training-free NAS.

2.2 Training-free Neural Architecture Search

Mellor et al. (2021a) presented a method for scoring a network architecture without any training and prior knowledge of trained network architectures. They focused on CNN architectures in the sample space of various NAS benchmarks, predicting the accuracy of the architectures on the CIFAR-10, CIFAR-100, and ImageNet image classification benchmarks. While Mellor et al.'s proposed method showed a correlation between their score and actual trained accuracy, it decreased with more complex datasets like ImageNet and architectures with high accuracy. Mellor et al. also found that the images chosen for the mini-batch and initialization weights of the model have negligible impact on their score. Their method predicted accuracies of architectures in seconds, and is easily combined with traditional NAS algorithms.

Abdelfattah et al. (2020) introduced a series of additional training-free metrics for CNNs with image classification tasks, based in network pruning literature, aiming to improve performance. They also tested on their metrics on other search spaces with different tasks, including NAS-Bench-NLP with RNNs and NAS-Bench-ASR, but found significantly reduced performance in these search spaces.

3 Training-free NAS Metrics

A series of training-free NAS metrics have been proposed in recent literature. These metrics look at specific aspects of an architecture, such as parameter gradients, activation correlations, and weight matrix rank. Most metrics can be generalized to any type of neural network, but have only been tested on CNN architectures. For transformer architectures, we also adapt various attention parameter pruning metrics as training-free metrics, scoring the entire network.

3.1 Synaptic Saliency

In the area of network pruning, Tanaka et al. (2020) proposed synaptic saliency, a score for approximating the change in loss when a specific parameter is removed. Synaptic saliency is based on the idea of preventing layer collapse while pruning a network, which significantly decreases the network's accuracy. Synaptic saliency is expressed by

$$S(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta, \tag{1}$$

where $\mathcal{L}$ is the loss function of the network, θ is the network's parameters, and $\odot$ is the Hadamard product. Abdelfattah et al. (2020) generalize synaptic saliency as a training-free metric for NAS by summing over all N parameters in the network: $S = \sum_{i=1}^{N} S(\theta_i)$. Abdelfattah et al. (2020) found that synaptic saliency slightly outperforms Jacobian covariance on NAS-Bench-201.

3.2 Jacobian Covariance

Jacobian Covariance is a training-free NAS metric for CNN networks proposed by Mellor et al. (2021b). Given a minibatch of input data, the metric assesses the Jacobian of the network's loss function with respect to the minibatch inputs, $\mathbf{J} = \left(\frac{\partial \mathcal{L}}{\partial x_1} \cdots \frac{\partial \mathcal{L}}{\partial x_N}\right)$. Further details of the metric can be found in the original paper.

Celotti, Balafrej, and Calvet (2020) expand on Jacobian Covariance with a series of variations on the metric, aiming to speed up computation and refine the metric's effectiveness. These include using cosine similarity instead of a covariance matrix to calculate similarity, expressed by

$$S = 1 - \frac{1}{N^2 - N} \sum_{i=1}^{N} \left| J_n J_n^t - I_n \right|^{\frac{1}{20}}, \tag{2}$$

where J_n is the normalized Jacobian and the minibatch has N inputs. They also add various noise levels to the input minibatch, hypothesizing that an architecture with high accuracy will be robust against noise.

3.3 Activation Distance

In a revised version of their paper, Mellor et al. (2021a) developed a metric that directly looks at the ReLU activations of a network. Given a minibatch of inputs fed into the network, the metric calculates the similarity of the activations within the initialized network between each input using the Hamming distance. Mellor et al. conclude that the more similar the activation map for a given set of inputs are to each other, the harder it is for the network to disentangle the representations of the inputs during training.

3.4 Synaptic Diversity

Zhou et al. developed a metric specific for vision transformers (ViT) (Dosovitskiy et al. 2021). Synaptic diversity is based upon previous research on rank collapse in transformers, where for a set of inputs the output of a multi-headed attention block converges to rank 1, significantly harming the performance of the transformer. Zhou et al. use the Nuclear-norm of an attention heads's weight matrix W_m as an approximation of its rank, creating a synaptic diversity score:

$$S_D = \sum_m \left|\left| \frac{\partial \mathcal{L}}{\partial W_m} \right|\right|_{nuc} \odot ||W_m||_{nuc}.$$

3.5 Hidden Covariance

We propose a new metric specific for RNNs, based on the hidden states between each layer of the RNN architecture. Previous NAS metrics focus on either the activation functions within an architecture, or all parameters of the architecture. The hidden state of an RNN layer encodes all of the information of the input, before being passed to the next layer or the final output. Similar to Mellor et al. (2021a), we hypothesize that if the hidden states of an architecture given a minibatch of inputs are similar to each other, the more difficult it would be to train the architecture.

Given the hidden state $\mathbf{H}(\mathbf{X})$ of a specific layer of the RNN with a minibatch of N inputs $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, observe the covariance matrix to be

$$\mathbf{C} = (\mathbf{H} - \mathbf{M_H})(\mathbf{H} - \mathbf{M_H})^T,$$

where $(\mathbf{M_H})_{ij} = \frac{1}{N}\sum_{n=1}^N \mathbf{H}_{in}$. Then, calculate the Pearson product-moment correlation coefficients matrix

$$\mathbf{R}_{ij} = \frac{\mathbf{C}_{ij}}{\sqrt{\mathbf{C}_{ii}\mathbf{C}_{jj}}}.$$

As with Mellor et al.'s Jacobian Covariance score, the final metric is calculated with the Kullback–Leibler divergence of the kernel of $\mathbf{R}$, which has the N eigenvalues $\lambda_1, \cdots, \lambda_N$:

$$S(\mathbf{H}) = -\sum_{n=1}^N \left(\log(\lambda_n + k) + \frac{1}{\lambda_n + k} \right),$$

where $k = 10^{-5}$.

3.6 Attention Confidence, Importance, and Softmax Confidence

For transformer-specific metrics, we look into current transformer pruning literature. Voita et al. (2019) propose pruning the attention heads of a trained transformer encoder block by computing the "confidence" of a head using a sample minibatch of input tokens. Confident heads attend their output highly to a single token, and, hypothetically, are more important to the transformer's task. Behnke and Heafield (2020) attempt to improve on attention confidence by looking at the probability distribution provided by an attention head's softmax layer. Alternatively, Michel, Levy, and Neubig (2019) look at the sensitivity of an attention head to its weights being masked, by computing the product between the output of an attention head with the gradient of its weights. These three attention metrics are summarized by:

$$\text{Confidence: } A_h(\mathbf{X}) = \frac{1}{N}\sum_{n=1}^N |\max(\text{Att}_h(\mathbf{x}_n))|$$

$$\text{Importance: } A_h(\mathbf{X}) = \left| \text{Att}_h(\mathbf{X}) \frac{\partial \mathcal{L}(\mathbf{X})}{\partial \text{Att}_h(\mathbf{X})} \right|$$

$$\begin{matrix}\text{Softmax}\\ \text{Confidence}\end{matrix}: A_h(\mathbf{X}) = \frac{1}{N}\sum_{n=1}^N |\max(\text{softmax}_h(\mathbf{x}_n))|$$

where $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ is a minibatch of N inputs, $\mathcal{L}$ is the loss function of the model, and Att_h and softmax_h are an attention head and its softmax respectively. We expand these metrics into an overall score for the entire network by summing over all attention heads: $\mathcal{A}(\mathbf{X}) = \sum_h \text{Att}_h(\mathbf{X})$.

4 Methods

4.1 NAS Benchmarks

Because of the large search space for neural architectures, it is challenging to have direct comparisons between various NAS algorithms. A series of NAS benchmarks (Mehta et al. 2022) have been created, which evaluate a set of architectures within a given search space and store the trained metrics in a lookup table. These benchmarks include NAS-Bench-101 (Ying et al. 2019), NAS-Bench-201 (Dong and Yang 2020), and NAS-Bench-301 (Siems et al. 2021) with CNNs for image classification, NAS-Bench-ASR with convolutional LSTMs for automatic speech recognition (Mehrotra et al. 2021), and NAS-Bench-NLP with RNNs for language modeling tasks (Klyuchnikov et al. 2022). Because all the architectures in a NAS benchmark have already been trained, they also allow for easier development of NAS algorithms without the large amounts of computational power required to train thousands of architectures. However, there are currently no NAS benchmarks for transformer or BERT-based architectures, likely due to the longer time and higher computational power to train transformers.

To evaluate training-free metrics on RNNs, we utilize the NAS-Bench-NLP benchmark (Klyuchnikov et al. 2022), which consists of 14,322 RNN architectures trained for language modeling with the Penn Tree Bank dataset. The architecture search space is defined by the operations within an RNN cell, connected in the form of an acyclic digraph. The RNN architecture consists of three identical stacked cells with an input embedding and connected output layer. In our evaluations, the NAS-Bench-NLP architectures which did not complete training in the benchmark or whose metrics could not be calculated for were discarded, leaving 8,795 architectures.

4.2 BERT Benchmark for NAS

Because no preexisting NAS benchmark exists for BERT-based models, we need to pretrain and evaluate a large set of various BERT architectures in order to evaluate our proposed training-free NAS metrics. Certain choices were made in order to speed up pretraining. These included: using the

Architecture Element	Allowed Hyperparameters
Hidden dimension	$\{128, 256\}$
Number of Encoder Layers	$\{2, 4\}$
Type of attention operator	{self-attention, linear transform, span-based dynamic convolution}
Number of operation heads	$\{2, 4\}$
Feed-forward dimension	$\{512, 1024\}$
Number of feed-forward stacks	$\{1, 3\}$
Attention operation parameters	
if self-attention	{scaled dot-product, multiplicative}
if linear transform	{discrete Fourier, discrete cosine}
if dynamic convolution	convolution kernel size: $\{5, 9\}$

Table 1: The FlexiBERT search space, with hyperparameter values spanning those found in BERT-Tiny and BERT-Mini. Hidden dimension and number of encoder layers is fixed across the whole architecture; all other parameters are heterogeneous across encoder layers. The search space encompasses 10,621,440 architectures.

ELECTRA pretraining scheme (Clark et al. 2020), choosing a search space consisting of small BERT architectures, and shortening pretraining. Once a set of optimal architectures is found using our metrics, it can be scaled up into a full-sized architecture comparative to state-of-the-art architectures.

BERT Search Space BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2019) consists of a series of encoder layers with multi-headed self-attention, taken from the original transformer model proposed by Vaswani et al. (2017). Numerous variations on the original BERT model have been developed. For our architecture search space, we utilize the FlexiBERT search space (Tuli et al. 2022), which has improvements over other proposed BERT search spaces. Foremost is that the encoder layers in FlexiBERT are heterogeneous, each having their own set of architecture elements. FlexiBERT also incorporates alternatives to the multi-headed self-attention into its search space. The search space is described in Table 1.

The architectures in the Flexibert search space are relatively small, as the hyperparameters in FlexiBERT search space spans those in BERT-Tiny and BERT-Mini (Turc et al. 2019). However, Kaplan et al. (2020) show many many attributes of a transformer architecture, including number of parameters, scale linearly with the architecture's performance. Thus, a transformer architecture can easily be scaled up by increasing its hyperparameter values equivalent to those found in larger architecture, in order to achieve greater performance. This methodology was utilized in EcoNAS algorithm (Zhou et al. 2020), which explores a reduced search space, before scaling up to produce the final model.

To allow for simpler implementation of the FlexiBERT search space and the utilization of absolute positional encoding, we keep the hidden dimension homogeneous across all encoder layers. In total, this search space encompasses 10,621,440 different transformer architectures.

ELECTRA Pretraining Instead of the traditional masked language modeling used to pretrain BERT-based models, we implemented the ELECTRA pretraining scheme (Clark et al. 2020), which uses a combination generator-discriminator model with a replaced token detection task. As the ELECTRA task is defined over all input tokens, instead of the masked tokens, it is significantly more compute efficient and results in better finetuning performance when compared to masked-language modelling. Notably, ELECTRA scales well with small amounts of compute, allowing for efficient pretraining of small BERT models.

Architecture Training and Evaluation We pretrain a random sample of 500 models from the FlexiBERT subspace using ELECTRA with the OpenWebText dataset, consisting of 38 GB of tokenized text data from 8,013,769 documents (Gokaslan and Cohen 2019). OpenWebText is based on OpenAI's WebText dataset (Radford et al. 2019). Pretraining occurs with TPUv2s with 8 cores and 64 GB of memory, using Google Collabortory. We finetune and evaluate the architectures on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al. 2019). The hyperparameters used for pretraining and finetuning are the same as those used for ELECTRA-Small. However, the sampled architectures were only pretrained for 100,000 steps for the best tradeoff benefit between pretraining time and GLUE score. All GLUE results are from the dev set.

5 Experimental Results of Training-free Metrics

For the training-free NAS metrics presented, we empirically evaluate how well the metric performs in predicting the trained performance of an architecture. We use Kendall rank correlation coefficient (Kendall τ) and Spearman rank correlation coefficient (Spearman ρ) to quantitatively evaluate the metrics by comparing them with the trained performance of the architectures within NAS-Bench-NLP and our BERT Benchmark.

5.1 Training-free Metrics for RNNs

We ran the training-free metrics on 8,795 architectures in NAS-Bench-NLP. A summary of our results are show in Figure 1. Computing these metrics was very efficient, only requiring a forward and backward pass with a single minibatch of sample data, in order to compute one set of gradients. Furthermore, all the metrics can be computed simultaneously on the same input and gradients.

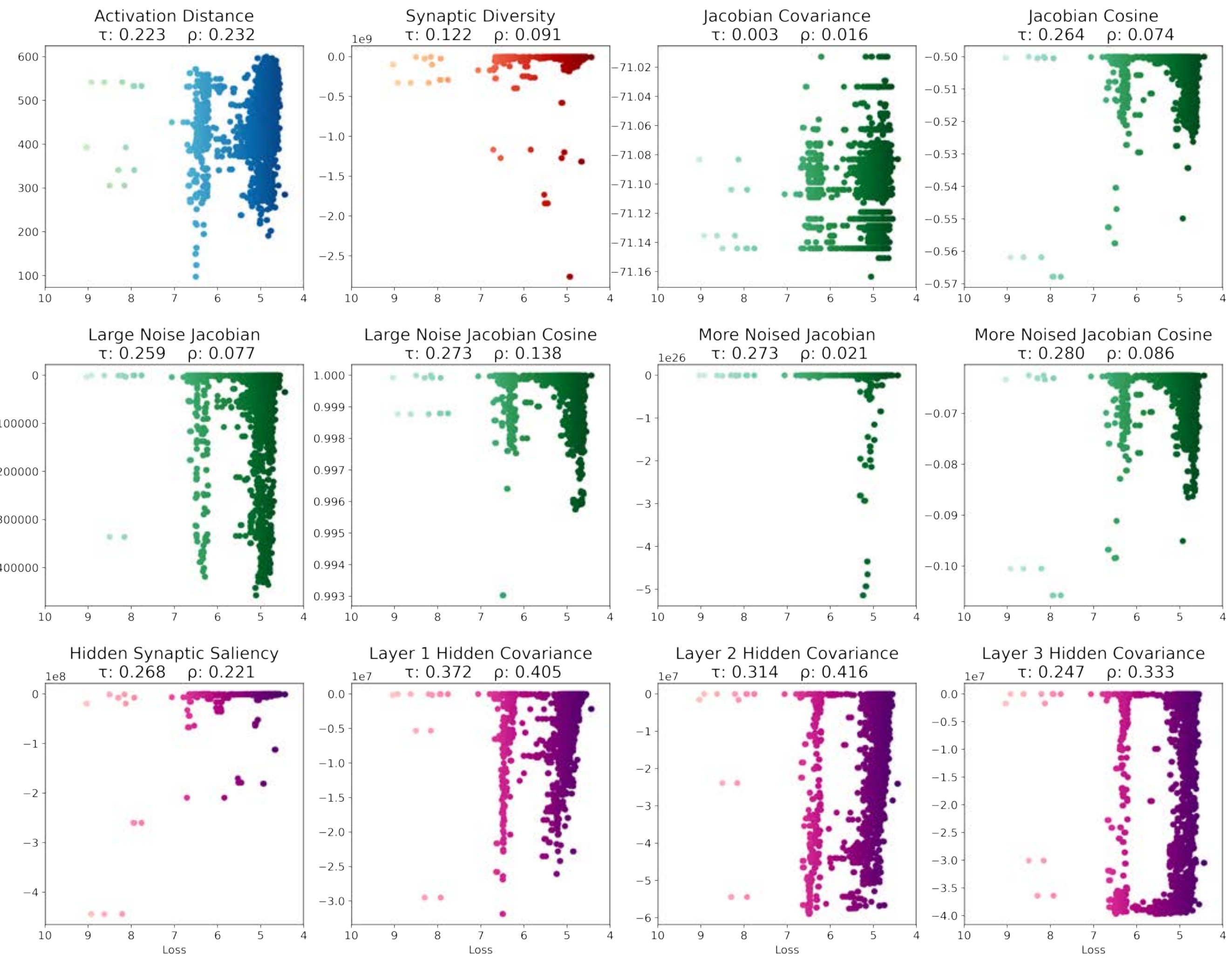


Figure 1: Plots of training-free metrics evaluated on 8,795 RNN architectures in NAS-Bench-NLP, against test loss of the architectures assessed on the Penn Tree Bank dataset when trained. Kendall τ and Spearman ρ also shown. Only our Hidden Covariance metric performed on the first and second layer of the RNN showed a substantial correlation between the metric and trained test loss. Some other metrics do have some positive correlation.

Most metrics preform poorly on predicting the loss of a trained RNN architecture, including all the existing training-free metrics designed for CNN architectures. None surpassed a Kendall τ value of 0.28. Our proposed Hidden Covariance score preforms the best out of all metrics, achieving a Kendall τ value of 0.3715. It is clear that the initialized hidden states of an RNN contain the most salient information for predicting the RNN's trained accuracy.

5.2 Training-free Metrics for BERT Architectures

We investigated the series of training-free metrics on our own NAS BERT benchmark of 500 architectures sampled from the FlexiBERT search space. Results are shown in Figure 2. Compared to their performance on NAS-Bench-NLP, all the training-free metrics, including our proposed metrics based on attention head pruning, performed poorly. Only the Attention Confidence metric had a significant positive correlation, with a Kendall τ of 0.27.

A notable reference point for training-free metrics is the number of trainable parameters in a transformer architecture. Previous research has shown a strong correlation between number of parameters and model performance across a wide range of transformer sizes and hyperparameters (Kaplan et al. 2020). Our NAS BERT Benchmark displays this same correlation (Figure 3). In fact, the Kendall τ value for number of parameters is 0.44, significantly surpassing all training-free metrics.

Great care must be used when developing training-free metrics to ensure that the metric is normalized for number of parameters or other high-level features of the network, such as number of layers or hidden size. In Zhou et al.'s proposed DSS-indicator score for vision transformers (a combi-

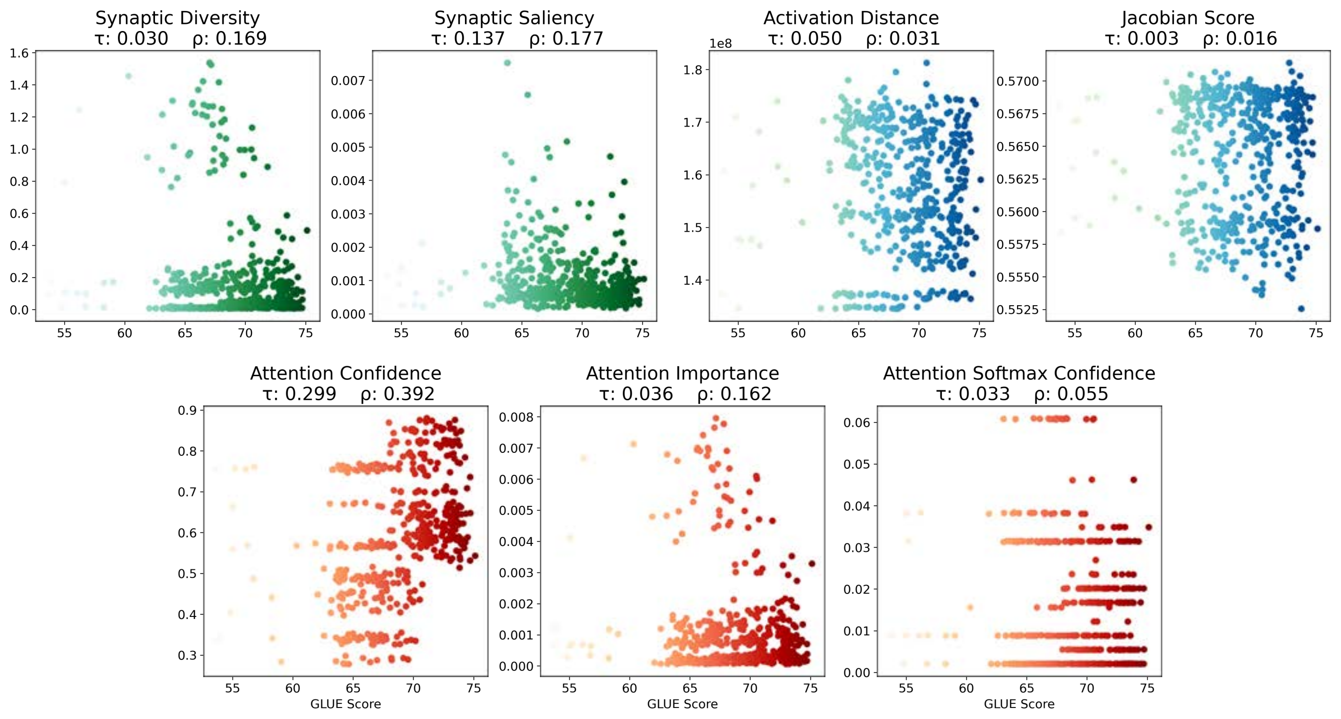


Figure 2: Plots of training-free metrics evaluated on 500 architectures randomly sampled from the FlexiBERT search space, against GLUE score of the pretrained and finetuned architecture. All metrics are normalized against number of features. Only our Attention Confidence metric displayed some positive correlation between the metric and final GLUE score.

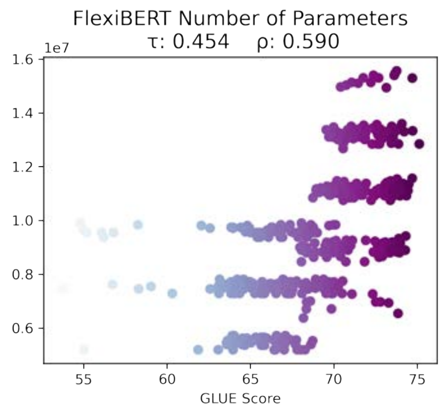


Figure 3: Correlation between number of parameters in a BERT-based architecture and its pretrained and fintuned GLUE score, for 500 architectures from the FlexiBERT search space. Number of parameters shows a strong correlation with architecture performance, substantially outperforms all training-free metrics evaluated.

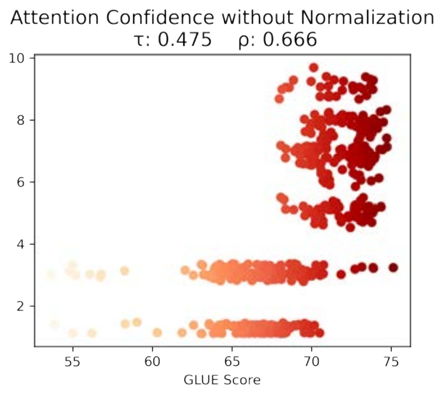


Figure 4: Attention Confidence metric evaluated on architectures from the FlexiBERT search space, without normalization for number of features. The metric's performance substantially improves when not normalized, and its plot mirrors that of number of parameters, as indicated by its Kendall τ value.

nation of synaptic saliency and synaptic diversity metrics), they did not normalize the score for the number of features in the network. Instead, the DSS-indicator almost directly corresponds to the number of parameters in an architecture, as shown in their figures, thus yielding their high Kendall τ of 0.70. We witnessed a similar pattern with our series of metrics. For our highest performing score, Attention Confidence, had a Kendall τ of 0.49 without normalization for number of features, comparable to number of parameters, but decreased to 0.30 with normalization (Figure 4).

6 Discussion

Neural architecture search for transformers is a fundamentally different task than neural architecture search for CNNs and RNNs. Almost all search spaces for transformers relies on the same fundamental paradigm of an attention module followed by a feed-forward module within each encoder/decoder block, connected linearly (Wang et al. 2020; Yin et al. 2021; Zhao et al. 2021). Conversely, most search spaces for CNNs and RNNs, including NAS-Bench-201 and NAS-Bench-NLP, use an cell-based method, typically with an acyclic digraph representing the connections between operations (Dong and Yang 2020; Jing, Xu, and Zugeng 2020; Klyuchnikov et al. 2022; Tan et al. 2019), allowing for significantly more flexibility in cell variation. For CNN and RNN search spaces, the connections between operations within a cell have a greater impact on the architecture's performance than number of parameters. In NAS-Bench-NLP, there is no correlation between number of parameters and model performance (Figure 5); hence, previous studies did not need to normalize their training-free metrics for number of parameters. Furthermore, we hypothesize that for transformer search spaces, the number of parameters in an architecture dominates the model performance, explaining the poor performance for training-free NAS metrics.

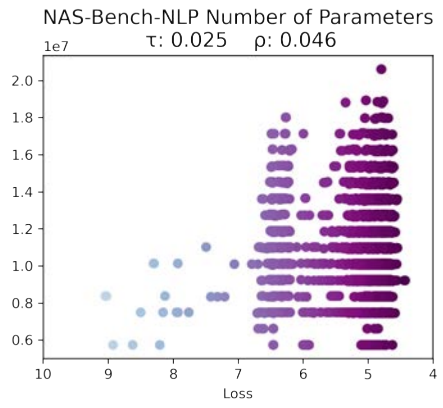


Figure 5: Plot of number of parameters against test loss for 8,795 RNNs architectures in NAS-Bench-NLP. Unlike the architectures in the FlexiBERT search space, there is no correlation between number of parameters and architecture performance for the architectures in NAS-Bench-NLP.

With this hypothesis, we propose an alternative to training-free metrics for current transformer neural architecture search, based upon transformer scaling laws. When model size and number of parameters is not a concern, increasing the size and dimensions of the architecture will consistently increase model performance. Thus, one should limit the transformer search space to larger model sizes in order to find better performing models more quickly.

However, it is often the case that the number of parameters within a model must be limited due to various computational limitations, including training time and cost or deployment on resource-constrained devices. First, one should set a targeted number of parameters for the architectures within the transformer search space. The ratios between various hyperparameters within the network can then be searched for with the NAS algorithm. Kaplan et al. (2020) found that model performance varies minimally between different hyperparameter ratios when number of parameters is fixed for architectures that are homogeneous between layers. They also present the most optimal ratios between hidden size, feed-forward dimension, number of layers, and number of attention heads. Therefore, increased focus should be placed on layer heterogeneity within the search space, with established hyperparameter ratios used as starting points.

While these suggestions can help with shrinking the search space for transformer architectures and speed up neural architecture search algorithms, they do not address the main problem regarding transformer architecture search: the inflexibility of current transformer search spaces. Unless transformer search spaces adopt the variability of connections provided by a cell-based methods, as used by CNN and RNN search spaces, simple heuristics such as number of parameters will be primary training-free predictor of transformer model performance. To our knowledge, two works have utilized a cell-based method for transformer search spaces, the original transformer architecture search paper, ”The Evolved Transformer,” by So, Le, and Liang, and its successor ”Primer” (So et al. 2021). Some research has been done with cell-based search spaces for Conformers (Shi et al. 2021) and Vision Transformers (Guo et al. 2020), but only on the convolution modules of the architectures. Ultimately, there is significant opportunity for growth regarding transformer architecture search, and with it training-free NAS metric for transformers.

7 Conclusion

In this paper, we presented and evaluated a series of training-free NAS metrics for RNN and BERT-based transformer architectures, trained on language modeling tasks. We developed new training-free metrics targeted towards specific architectures, hidden covariance for RNNs and three metrics based on attention head pruning for transformers. We first verified the training-free metrics on with NAS-Bench-NLP, and found our hidden covariance metric outperforms existing training-free metrics on RNNs. We then developed our own NAS benchmark for transformers within the FlexiBERT search space, utilizing the ELECTRA scheme to significantly speed up pretraining. Evaluating the training-free metrics on our benchmark, our proposed Attention Confidence metric performs the best. However, the current search space paradigm for transformers is not well-suited for training-free metrics, and the number of parameters within a model is the most significant predictor of transformer performance. Our research shows that training-free NAS metrics are not universally successful across all architectures, and better transformer search spaces must be developed for training-free metrics to succeed. We hope that our work is a foundation for further research into training-free metrics for RNNs and transformers, in order to develop better and more efficient NAS techniques.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Abdelfattah, M. S.; Mehrotra, A.; Dudziak, L.; and Lane, N. D. 2020. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations 2021*.

Behnke, M.; and Heafield, K. 2020. Losing Heads in the Lottery: Pruning Transformer Attention in Neural Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2664–2674. Online: Association for Computational Linguistics.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Celotti, L.; Balafrej, I.; and Calvet, E. 2020. Improving Zero-Shot Neural Architecture Search with Parameters Scoring. Https://openreview.net/forum?id=4QpDyzCoH01.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. ArXiv:2003.10555 [cs].

Deng, B.; Yan, J.; and Lin, D. 2017. Peephole: Predicting Network Performance Before Training. ArXiv:1712.03351v1.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [cs].

Dong, X.; and Yang, Y. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Ninth International Conference on Learning Representations*.

Gokaslan, A.; and Cohen, V. 2019. OpenWebText Corpus.

Guo, Y.; Zheng, Y.; Tan, M.; Chen, Q.; Chen, J.; Zhao, P.; and Huang, J. 2020. NAT: Neural Architecture Transformer for Accurate and Compact Architectures. ArXiv:1910.14488 [cs, stat].

Istrate, R.; Scheidegger, F.; Mariani, G.; Nikolopoulos, D.; Bekas, C.; and Malossi, A. C. I. 2019. TAPAS: Train-Less Accuracy Predictor for Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 3927–3934. ArXiv:1806.00250v1.

Jing, K.; Xu, J.; and Zugeng, H. X. 2020. NASABN: A Neural Architecture Search Framework for Attention-Based Networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–7. ISSN: 2161-4407.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. ArXiv:2001.08361 [cs, stat].

Klyuchnikov, N.; Trofimov, I.; Artemova, E.; Salnikov, M.; Fedorov, M.; Filippov, A.; and Burnaev, E. 2022. NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing. *IEEE Access*, 10: 45736–45747. ArXiv:2006.07116v1.

Mehrotra, A.; Ramos, A. G. C. P.; Bhattacharya, S.; Dudziak, L.; Vipperla, R.; Chau, T.; Abdelfattah, M. S.; Ishtiaq, S.; and Lane, N. D. 2021. NAS-Bench-ASR: Reproducible Neural Architecture Search for Speech Recognition. In *International Conference on Learning Representations*.

Mehta, Y.; White, C.; Zela, A.; Krishnakumar, A.; Zabergja, G.; Moradian, S.; Safari, M.; Yu, K.; and Hutter, F. 2022. NAS-Bench-Suite: NAS Evaluation is (Now) Surprisingly Easy. In *Ninth International Conference on Learning Representations*. ArXiv:2201.13396v2.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2021a. Neural Architecture Search without Training. In *Proceedings of the 38th International Conference on Machine Learning*, 7588–7598. ArXiv:2006.04647v3.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2021b. Neural Architecture Search without Training.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Palangi, H.; Deng, L.; Shen, Y.; Gao, J.; He, X.; Chen, J.; Song, X.; and Ward, R. 2016. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4): 694–707. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; and others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J.; and others. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. ArXiv:1802.01548 [cs].

Shi, X.; Zhou, P.; Chen, W.; and Xie, L. 2021. Efficient Gradient-Based Neural Architecture Search For End-to-End

ASR. In *Companion Publication of the 2021 International Conference on Multimodal Interaction*, ICMI '21 Companion, 91–96. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8471-1.

Siems, J. N.; Zimmer, L.; Zela, A.; Lukasik, J.; Keuper, M.; and Hutter, F. 2021. NAS-Bench-301 and the Case for Surrogate Benchmarks for Neural Architecture Search.

So, D.; Le, Q.; and Liang, C. 2019. The Evolved Transformer. In *Proceedings of the 36th International Conference on Machine Learning*, 5877–5886. ArXiv:1901.11117v4.

So, D.; Mańke, W.; Liu, H.; Dai, Z.; Shazeer, N.; and Le, Q. V. 2021. Searching for Efficient Transformers for Language Modeling. In *Advances in Neural Information Processing Systems*, volume 34, 6010–6022. Curran Associates, Inc.

Strubell, E.; Ganesh, A.; and McCallum, A. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ArXiv:1906.02243v1.

Sundermeyer, M.; Schlüter, R.; and Ney, H. 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2815–2823. Long Beach, CA, USA: IEEE. ISBN 978-1-72813-293-8.

Tan, M.; and Le, Q. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 6105–6114. ArXiv:1905.11946v5.

Tanaka, H.; Kunin, D.; Yamins, D. L.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, 6377–6389. Curran Associates, Inc.

Tuli, S.; Dedhia, B.; Tuli, S.; and Jha, N. K. 2022. FlexiBERT: Are Current Transformer Architectures too Homogeneous and Rigid? ArXiv:2205.11656 [cs].

Turc, I.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. ArXiv:1908.08962 [cs].

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; and Titov, I. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5797–5808. Florence, Italy: Association for Computational Linguistics.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. ArXiv:1804.07461 [cs].

Wang, H.; Wu, Z.; Liu, Z.; Cai, H.; Zhu, L.; Gan, C.; and Han, S. 2020. HAT: Hardware-Aware Transformers for Efficient Natural Language Processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7675–7688. Online: Association for Computational Linguistics.

Yin, Y.; Chen, C.; Shang, L.; Jiang, X.; Chen, X.; and Liu, Q. 2021. AutoTinyBERT: Automatic Hyper-parameter Optimization for Efficient Pre-trained Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 5146–5157. Online: Association for Computational Linguistics.

Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; and Hutter, F. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *Proceedings of the 36th International Conference on Machine Learning*, 7105–7114. PMLR. ISSN: 2640-3498.

Yu, Y.; Si, X.; Hu, C.; and Zhang, J. 2019. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7): 1235–1270.

Zhao, Y.; Dong, L.; Shen, Y.; Zhang, Z.; Wei, F.; and Chen, W. 2021. Memory-Efficient Differentiable Transformer Architecture Search. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 4254–4264.

Zhou, D.; Zhou, X.; Zhang, W.; Loy, C. C.; Yi, S.; Zhang, X.; and Ouyang, W. 2020. EcoNAS: Finding Proxies for Economical Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11396–11404.

Zhou, Q.; Sheng, K.; Zheng, X.; Li, K.; Sun, X.; Tian, Y.; Chen, J.; Ji, R.; and Laboratory, P. C. 2022. Training-free Transformer Architecture Search. In *Proceedings of the 2022 IEEE/CVF Computer Vision and Pattern Recognition Conference*. ArXiv:2203.12217v1.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations*. ArXiv.1611.01578v2.

Light Weight Transforms Ramp up Autonomy of UAVs

Raymond Dueñas
University of Colorado Colorado Springs
duenas2100@gmail.com

Adham Ayabi
University of Colorado Colorado Springs
aatyabi@uccs.edu

Abstract

Autonomous uncrewed areal vehicles require the ability to navigate various environments without collision failures. These systems already serve important roles in a variety of fields ranging from entertainment to military application. There is a desire to replace costly multi-sensor based systems with a system based solely on computer vision. However, these systems suffer from varying accuracy in object recognition and in some cases object recognition is to slow to avoid a collision failure. Currently the best solutions for computer vision based systems implement artificial neural networks or an algorithm written for a specific task. Based on (Wang et al. 2022) which illustrate the high accuracy of OFA this work implements OFA as an alternative to artificial neural networks and in doing so expects to produce a vision based autonomous system with minimal parameters high accuracy and competitive speed.

Introduction

Uncrewed arial vehicles (UAVs) have a wide range of applications, from children's toys to military operations. One such environment is UAV racing as a sport which has produced championship UAV pilots that can fly UAVs through obstacle courses performing maneuvers including corkscrews, loops, suicide dives, and reaching speeds up to 120mph. There is a high demand for advancement in developing an autonomous UAV (AUAV) system with the ability to navigate through obstacles, avoid collisions, and reliably execute objectives. The UAV maneuverability achieved by champion UAV racing pilots serves as a benchmark for AUAVs, a benchmark with an extreme gap in performance. Lockheed Martin and CEO of The Drone Racing League, Nicholas Horbaczewski shared a vision of leveraging the sport to ramp up advancements in AUAV technology. This vision brought about the foundation of Lockheed Martin's AlphaPilot Innovation Challenge which challenges participants to design AUAV capable of piloting through professional drone racing courses. The metrics used to determine the winning AUAV are, firstly, the percentage of course completed, and secondly, the speed of completion. The AlphPilot Innovation Challenge finalist competed at Artificial Intelligence Robotic Racing (AIRR) World Championship. At this event, of the nine finalists, only two AUAVs successfully completed the course, a failure rate of 78%. Of the two AUAVs that completed the course, the winner had an average speed of 1.5m/s. The architecture utilized by this AUAV serves as a benchmark for this work. The Game of Drones is another drone racing platform and is where the second benchmark for this work is derived. A Microsoft initiative working to close the gap between AUAVs and piloted UAVs The Game of Drones runs on the AirSim virtual platform developed by Microsoft and Stanford. Teams load there computer vision-based navigation models onto a virtual drone that then uses the model to traverse a rigorous virtual course. Winning requires completing more gates than any other team or completing the same number of gates with a faster track time. This work utilizes recent advances in lightweight visual transformers. Applying the visual transformer model OFP a sequence to sequence framework presented by (Wang et al. 2022) in conjunction with the Separable Pyramidal Pooling EncordEr-Decoder (SPEED) presented by (Papa et al. 2022) for depth perception and object avoidance guided controllers based on (Zhang et al. 2020) work with monocular trajectory planning. Combining the three listed methods will yield a general solution for the computer vision based AUAV navigation situation. Providing enhanced object detection, route navigation, collision avoidance, and an increased success rate of monocular AUAV flight.

Related Work

Autonomous flight requires the successful syntheses of multiple dynamic objectives and systems. In this work we focus in on object detection, route estimation or obstacle avoidance and depth estimation. The following related works represent the top performing AUAV systems from three separate competitions and presents the methods utilized by each team with respect to the noted systems of focus for this work.

UZH Robotics and Perception Group: Optimal Methods meet Deep Learning for Autonomous Drone Racing

Object detection: Utilizing A deep network the team first delivers an input image to a shallow DroNet architecture based Convolutional Neural Network, the outputted caricaturists are then handled by two individual multilayer perceptrons.

Collision avoidance and Rout Estimation: Utilizing a successive two stage system, first a waypoint is derived from gate estimated location and a favorable path is chosen. In the next stage the onboard controller is relayed directions to navigate to the waypoint and flight path is tracked for improved stabilization between waypoints.
Depth Estimation: This model calculates the deep network derived regression of the input RGB image's mean in order to determine distance to a gate.

Sejong University: Report for Game of Drones A NeurIPS 2019 Competition

Object detection:: Using a Neural Network as an object detection model. The team implements U-Net segmentation an actor net and a critic net in the process of training the neural network. In order to develop a reward based guidance for navigation decisions derived from the initial detection of a gate and current estimated AUAV location.
Collision avoidance and Rout Estimation: Developing and implementing a rule-based control scheme called moveBySplineAsnc and moveOnSplineAsnc. The control scheme was trained by running the actor net through a virtual gate from a number of approach trajectories and presenting it with a risk reward value depending on if it makes it through the gate or suffers a collision failure while the segmentation compressed and preserved valuable data gained from the simulation.
Depth Estimation: (At this point it is unclear to me how they handle depth estimation. I believe it is derived from data gained from their object detection model)

MAVLab: A Computationally Efficient Vision-Based Navigation And Control Strategy

Object detection: The system utilizes deep learning based optic flow and algorithms they call Snake gate detection and Histogram gate algorithm which were uniquely designed for the competition.
Collision avoidance and Rout Estimation: Utilizing a PD controller and the Snake gate detection algorithm the AUAV is centered to the gate when ever there is a positive reading of a gate in view. To compensate for situations when no gate is in view the team implements a state estimator arc to turn the drone in the direction of the next gate.
Depth Estimation: Provided attitude estimate to the Snake Gate algorithm developed, also allows for depth estimate.

MAVLab, the winners of the AlphPilot Innovation Challenge, lay out the process of developing their benchmark AUAV in (Li et al. 2020) . A detailed system overview establishes that the system hardware utilized consists of a camera with six optical elements and 14 Megapixels sensor, Parrot p7 dual-core CPU cortex 9 (max 2GHZ), an MPU 6050 IMU and sonar with less than 8m range. Their AUAV utilizes a novel snake gate detection algorithm to identify and a PD controller to steer the drone to the center of the detectable rectangular-shaped gates. Utilizing classic complementary filter for attitude and heading reference systems (AHRS) then using Kalman filter to fuse AHRS and IMU measurements to estimate position. The system implemented a prediction-based feed-forward control scheme when the steer when snake gate detection algorithm does not detect a gate. Lastly, as a low-level attitude controller, their system employed an adaptive incremental nonlinear dynamic inversion (INDI). Utilizing the described AUAV was able to navigate a course at an average speed of 1.5m/s. However, (Li et al. 2020) explains that there are failure cases where the drone crashes into the gate due to late gate detection and complete detection failures.

Problem Statement

Working to develop a robust AUAV is a global effort. Currently, the success rate in developing an AUAV system that can navigate a course without crashing is at most 22% when considering the AIRR world championships, which–within the scope of the competition–consist of the most competitive AUAVs yet to be developed. This work aims to produce a general AUAV system that efficiently detects and classifies objects, develops superior navigational routes, and significantly reduces collision rates, all while increasing the AUAV's rate of travel. Lastly, the systems utilized by the benchmark AUAV employed a visual navigation algorithm specific to rectangular gate detection and will fail if the gate shape is changed. The successful execution of this work will produce a general system that does not rely on a mission specific algorithm, and instead accepts situational parameters to yield a significant gain in the operational scope of the AUAV.

Approach

Proposed System

To handle object detection, the lightweight vision transformer, OFA, which through a sequence-to-sequence learning framework, can perform vision and language tasks with state-of-the-art accuracy and competitive speed presented by (Wang et al. 2022) will be implemented. This transformer detects, classifies, provides objects in frame location, and can perform impressive image infilling. If this transformer can be implemented onto memory-constrained drones and tuned to increase throughput, these capabilities would provide a highly effective object detection model. In managing depth estimation and obstacle avoidance issues associated with monocular vision, this work proposes utilizing a paired CNN architecture, an overview of which can be seen in figure 1. The first CNN takes in a sequence of two one-dimensional frames that have been combined into one two denominational array and will produce optical flow-related values. The second CNN takes in the values produced by the first CNN and three-dimensional destination coordinate from which it outputs an optimal directional decision. The proposed CNN architecture aims to be as accurate as the RT-ViT model developed by (Ibrahem, Salem, and Kang 2022) and as fast as the SPEED model developed by (Papa et al. 2022). RT-ViT addresses depth estimation in real-time situations when depth estimation must be conducted with only monocular data. The performance of RT-ViT, reached state-of-the-art accuracy on multiple data

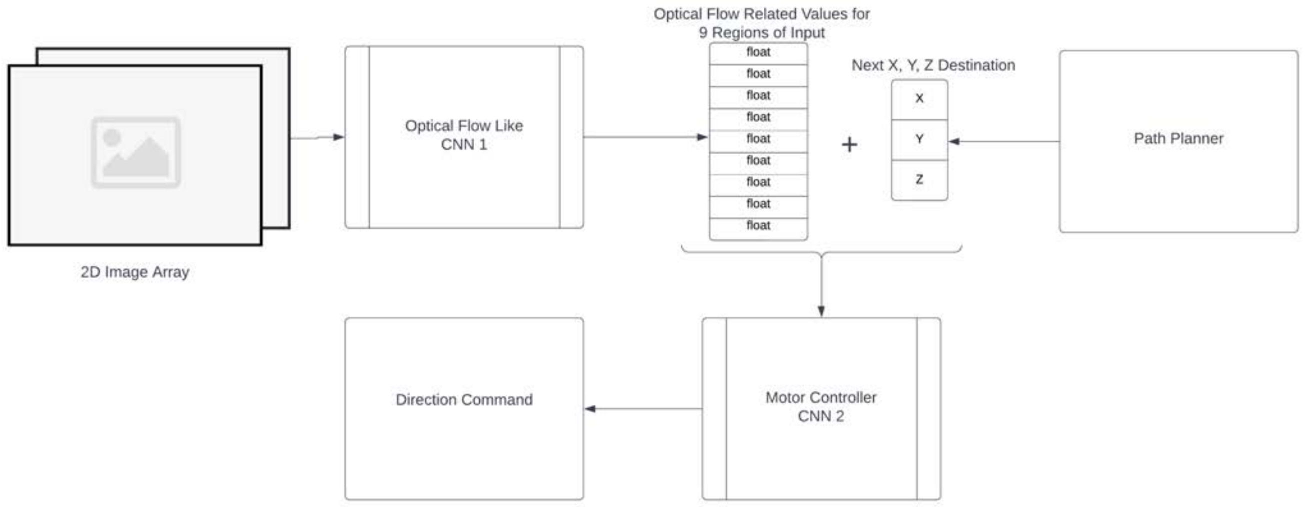


Figure 1: Paired CNN Architecture

sets, including NYU-depthv2 and CITYSCAPES.
However, the fastest RT-ViT model, ViT-t16+DE, had a maximum frame rate of 20.83. SPEED addresses collision avoidance in real-time situations, with only monocular data available. The performance of SPEED is better than other fast throughput architectures, even on low-resource settings (Papa et al. 2022). The SPEED model utilizes two depth-wise separable pyramidal pooling layers, increasing the inference frequency and reducing computational complexity. Utilizing NYU Depth v2 and DIML Kinectv2 datasets to benchmark monocular depth estimation. SPEED achieves state-of-the-art results for fast throughput compared with related works on the DIML Kinect v2 data set and outstanding results in error estimation compared to more complex models. When presented in 2019, SPEED's performance on the NYU Depth v2 data set was near the state-of-the-art at the time(Papa et al. 2022).

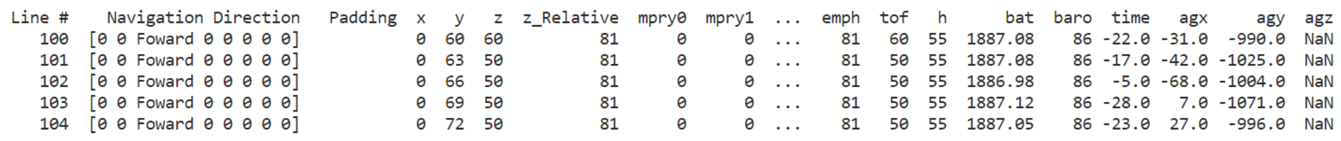
Furthermore, the obstacle avoidance scheme proposed by (Zhang et al. 2020) will be employed to provide reliable route estimation. The scheme will be implemented to develop the three-dimensional coordinates, which will be passed to the second CNN in the proposed paired CNN architecture. In their work (Zhang et al. 2020) states that reliable collision prevention estimation is unattainable with monocular data alone. Their work proposes an obstacle collision avoidance trajectory planning scheme as an alternative to collision prevention. Considering the characteristics of monocular optical measurement, they utilize two obstacle localization models based on relative range and relative angle. This model enhances the capability of AUAVs to avoid collision trajectories and achieve favorable results when compared with methods capable of geometric collision avoidance utilizing global knowledge.

Measuring Results

Environmental constraints make reproducing the MAVLab benchmark AUAV experiment unfeasible in this work. However, utilizing The Tello EDU model number: TLW004, this work will fit the benchmark model as tight as possible to the TLW004. Available specifications show that the TLW004 is equipped with 720p HD transmission, 5MP photos, FOV: 82.6, video: HD720P30, Intel processor, range finder, and barometer. After fitting the benchmark model to the available TLW004 the MAVLab metric tests will be run to establish a benchmark figure running MAVLab system on the TLW004. Once the benchmark has been established, the system proposed in this work will be fitted to the TLW004 and the tests will be repeated. Taking the percentage of course completed, average observed speed, gate detection hit/miss rate, and error distribution between estimated states and ground-truth states as metrics to compare the results of the proposed system with the established benchmark.

Microsoft's drone racing simulator, AirSim, will be employed in measuring this works proposed system against the top-performing system utilized by Sejoung University. Sejoung University provides metrics for the performance of their AUAV system in (Shin, Kang, and Kim). Loading the proposed work into the AirSim testing the performance of the system and measuring results with respect to the metrics provided by (Shin, Kang, and Kim) will allow for determination of the performance of the proposed system against that of the top-performing Sejoung University system.

After metric comparisons are complete, to test the scope of the proposed system's operational environment, the AUAV will navigate through three additional variations of a test course. The first variation will replace all rectangular gates with circular ones. The second will replace all circu-



```
Line #    Navigation Direction   Padding  x   y   z  z_Relative  mpry0  mpry1  ...  emph  tof   h      bat  baro  time   agx     agy  agz
   100  [0 0 Foward 0 0 0 0 0]            0  60  60          81      0      0  ...    81   60  55  1887.08    86 -22.0 -31.0  -990.0  NaN
   101  [0 0 Foward 0 0 0 0 0]            0  63  50          81      0      0  ...    81   50  55  1887.08    86 -17.0 -42.0 -1025.0  NaN
   102  [0 0 Foward 0 0 0 0 0]            0  66  50          81      0      0  ...    81   50  55  1886.98    86  -5.0 -68.0 -1004.0  NaN
   103  [0 0 Foward 0 0 0 0 0]            0  69  50          81      0      0  ...    81   50  55  1887.12    86 -28.0   7.0 -1071.0  NaN
   104  [0 0 Foward 0 0 0 0 0]            0  72  50          81      0      0  ...    81   50  55  1887.05    86 -23.0  27.0  -996.0  NaN
```

Figure 2: Example of drone state data collected.

lar gates with rectangular obstacles that must be maneuvered around to avoid a collision. Lastly, the course will combine rectangular gates, circular gates, rectangular obstacles, and circular obstacles. The successful completion of these differing environments will demonstrate a degree of the scope for the operational environment provided by the proposed AUAV system. Possible data sets for this work include Wild-UAV, EuRoC MAV, TUM monoVO, NYU Depth V1/V2, RGB+D, PASCAL VOC12, MS COCO, ImageNet, Open Images V6, and a self-derived data set for control outputs.

Experiments and Results

Data Set

TThe development of a data set was necessarily added to the scope of this work to experiment and train the proposed paired CNN architecture. Data was collected by flying the Tello drone indoors in both congested and clear environments. The video feed was recorded at a rate of 30 frames per second and stored as an avi file for each session. During each flight session, the drone state information was also recorded at a rate of 30 states per second. This information includes drones x, y, z, acceleration values relative height, and more. An example of this data can be seen in figure 2. While the drone state information includes the drone's height, z coordinate, it does not include x and y coordinates. Using the drone's tunable travel rate in centimeters per second measurements were conducted to calibrate the drones in-flight x and y coordinates relative to its initial hover position after take-off. This information was added to the drone's state data and can be seen in figure 2 labeled as x, y, z, and z-relative. Z-relative is the drone's height relative to anything directly beneath it, while z is in reference to initial take-off. With this information, odometry maps were developed and stored for reference as visual representations of the drone's traveled path. Lastly, after flight sessions were completed, a folder for each flight containing the individual frames from each session recording was created. From these frame files, a data set was generated containing the optical flow information pairs of frames. The results for each optical flow calculation were further processed, parsing each result into nine non-lapping regions, each of which was reduced to a single floating point value. The frames utilized in the optical flow operation and nine region representing floating point values generated are aligned in the data set generated. An example of this data and a visual representation of the nine optical flow regions can be seen in Figures 3 and 4.

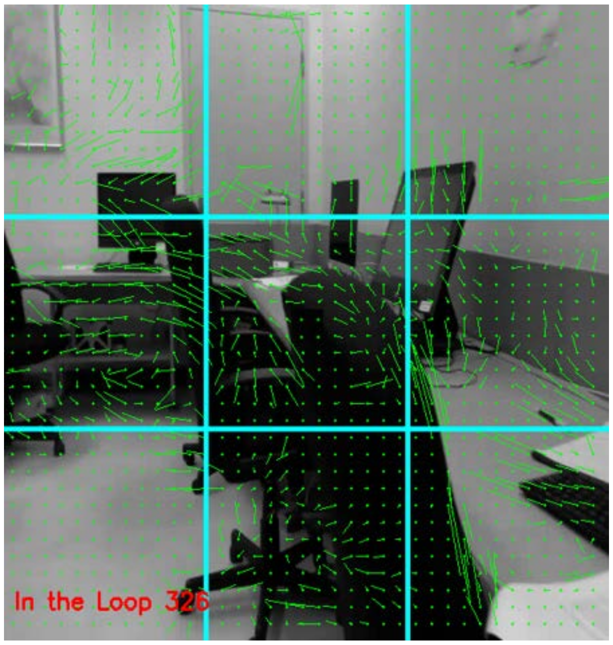


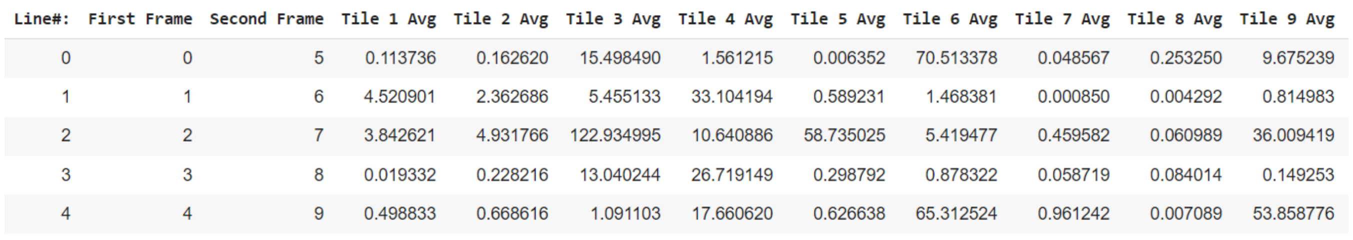
Figure 3: Visualisation of the nine optical flow region.

Model Development

The development of the paired CNN architecture has been split into two phases. The first of which develops the CNN in charge of taking in two sequential one-dimensional images paired together as one two-dimensional array and generating a nine-value representation of optical flow for the inputted sequence. After completion of this model, the second phase of model development would begin in which the model outputs drone motor control commands from the nine outputs of the first CNN and three denominational destination coordinate.

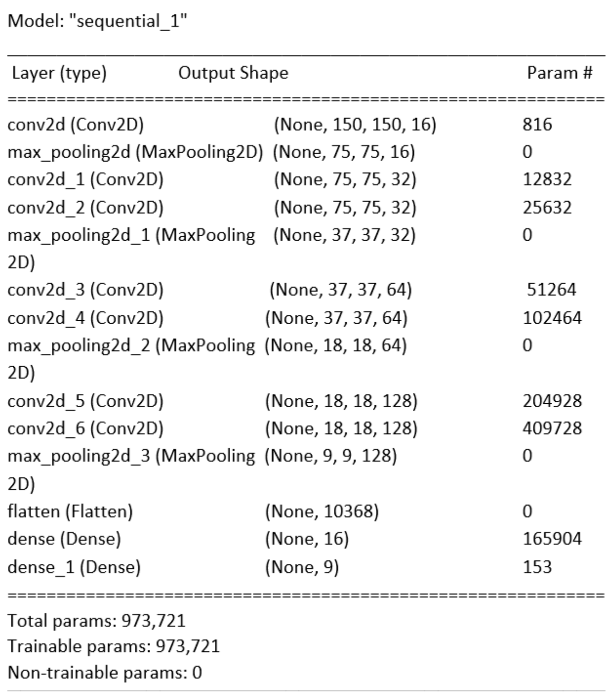
Results

In phase one, training the CNN took place in Colab Pro+ utilizing, Tensorflow, Keras, and Sklearn. The model would be required to take in an image as a variable and predict nine continuous values. From this, it was determined that this was a regression model task. As such mean squared error was implemented as the loss function, and mean absolute error was implemented as the metric for measuring the prediction error of the model. Experimenting with multiple architectures variations of the regression model and utilizing K-Fold cross-validation to provide the entirety of the data set, the



Line#:	First Frame	Second Frame	Tile 1 Avg	Tile 2 Avg	Tile 3 Avg	Tile 4 Avg	Tile 5 Avg	Tile 6 Avg	Tile 7 Avg	Tile 8 Avg	Tile 9 Avg
0	0	5	0.113736	0.162620	15.498490	1.561215	0.006352	70.513378	0.048567	0.253250	9.675239
1	1	6	4.520901	2.362686	5.455133	33.104194	0.589231	1.468381	0.000850	0.004292	0.814983
2	2	7	3.842621	4.931766	122.934995	10.640886	58.735025	5.419477	0.459582	0.060989	36.009419
3	3	8	0.019332	0.228216	13.040244	26.719149	0.298792	0.878322	0.058719	0.084014	0.149253
4	4	9	0.498833	0.668616	1.091103	17.660620	0.626638	65.312524	0.961242	0.007089	53.858776

Figure 4: Example of Optical flow values representing nine regions of optical flow.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)             Output Shape                  Param #
=================================================================
conv2d (Conv2D)                (None, 150, 150, 16)      816
max_pooling2d (MaxPooling2D)   (None, 75, 75, 16)        0
conv2d_1 (Conv2D)              (None, 75, 75, 32)        12832
conv2d_2 (Conv2D)              (None, 75, 75, 32)        25632
max_pooling2d_1 (MaxPooling    (None, 37, 37, 32)        0
2D)
conv2d_3 (Conv2D)              (None, 37, 37, 64)         51264
conv2d_4 (Conv2D)              (None, 37, 37, 64)        102464
max_pooling2d_2 (MaxPooling    (None, 18, 18, 64)        0
2D)
conv2d_5 (Conv2D)              (None, 18, 18, 128)       204928
conv2d_6 (Conv2D)              (None, 18, 18, 128)       409728
max_pooling2d_3 (MaxPooling    (None, 9, 9, 128)         0
2D)
flatten (Flatten)              (None, 10368)             0
dense (Dense)                  (None, 16)                165904
dense_1 (Dense)                (None, 9)                 153
=================================================================
Total params: 973,721
Trainable params: 973,721
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Architecture of optical flow value generating model.

lowest mean absolute error observed was 6.1520. Meaning that, on average, the model is 6.1520 units away from the correct prediction. The model architecture used to produce these results can be seen in figure 5.

Conclusion

The general solution presented in this work utilized the visual transformer model OFA and the proposed paired CNN architecture in conjunction with trajectory modeled for depth perception, obstacle avoidance, and motor controllers. The implementation of the first phase in model development has produced a model with prediction error that encourages improvement. Future work will include the implementation of phase two of model development, creating a model capable of producing optimal motor control commands towards its given destination. Completion of model development will prompt the highly anticipated testing of the proposed general solution against the benchmark AUAV models with respect to the defined metrics. Given the test results, further work to improve the model or further testing for robustness may be implemented..

References

Ibrahem, H.; Salem, A.; and Kang, H.-S. 2022. Rt-vit: Real-time monocular depth estimation using lightweight vision transformers. *Sensors* 22(10).

Li, S.; Ozo, M. M.; De Wagter, C.; and de Croon, G. C. 2020. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *Robotics and Autonomous Systems* 133:103621.

Papa, L.; Alati, E.; Russo, P.; and Amerini, I. 2022. Speed: Separable pyramidal pooling encoder-decoder for real-time monocular depth estimation on low-resource settings. *IEEE Access* 10:44881–44890.

Shin, S.-Y.; Kang, Y.-W.; and Kim, Y.-G. Report for game of drones: A neurips 2019 competition.

Wang, P.; Yang, A.; Men, R.; Lin, J.; Bai, S.; Li, Z.; Ma, J.; Zhou, C.; Zhou, J.; and Yang, H. 2022. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*.

Zhang, Z.; Cao, Y.; Ding, M.; Zhuang, L.; and Tao, J. 2020. Monocular vision based obstacle avoidance trajectory planning for unmanned aerial vehicle. *Aerospace Science and Technology* 106:106199.

The Effects of Subject Transfer on Transformer-Based EEG Classification of Finger Movement

Zach Snow
University of Kentucky
Lexington, Kentucky 40506
zsn222@uky.edu

Adham Atyabi
University of Colorado, Colorado Springs
1420 Austin Bluffs Parkway
Colorado Spring, Colorado 80918
aatyabi@uccs.edu

Abstract

Brain-Computer Interfaces (BCIs) employing Electroencephalogram (EEG) signals are powerful mechanisms for controlling prosthesis without physical manipulation. Current methods of processing EEG signals are either limited in their accuracy or in their flexibility, both of which hinder their broad use and application. EEG signals have exceptional temporal resolution but are comparatively lacking in spatial resolution which can make deep features of their signals difficult to interpret. Vision Transformers (ViTs) specialize in extracting patterns and features in images and categorizing them by detecting the importance of image regions in relation to other regions. This functionality makes ViTs strong candidates for translating EEG signals into specific motor functions. Transfer learning is the act of using a model trained on a similar task as a baseline for the intended task to reduce the amount of data required and improve classification accuracy. This paper proposes the use of Transformers and subject transfer to more effectively classify EEG signals as movements of individual fingers.

1 Introduction

An Electroencephalogram (EEG) is a brain signal recording method that uses multiple electrodes placed on the scalp to take measurements of signals in different areas of the brain. EEGs are often used to evaluate and diagnose neural conditions such as epilepsy, sleep disorders, and other encephalopathies. Brain-Computer Interfaces (BCIs) are systems that take signals from the brain, analyze them, and relay them as commands to an output device. One increasingly prevalent application of BCIs is in the field of prosthetics as BCIs have the capacity to convert brain signals from imagined motion into physical motion of a controlled device.

EEGs have high temporal resolution, typically taking samples of brain activity at a sampling rate of between 200 and 1000 times per second. The drawback of EEGs in prosthetics is their poor spatial resolution (Ieracitano et al. 2021). Current EEG systems consist of between 4 and 256 electrodes with the most common layouts having between 20 and 64 electrodes in their placement (Towle et al. 1993). The International standard 10-20 EEG system is pictured in Figure 1. Even with 256 electrodes, it is impossible to measure brain activity with any level of spatial precision. Since the number of neurons in a human brain is several orders of magnitude higher than the number of electrodes in a typical EEG, EEGs can only detect the activity of large clusters of neurons which leads to results that can be challenging to interpret.

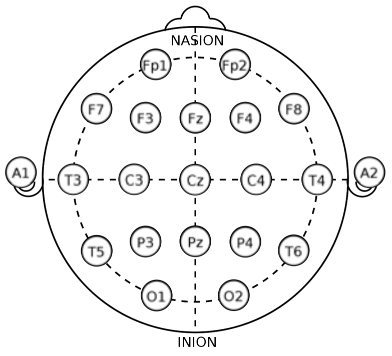


Figure 1: Electrode locations of the 10-20 system (Towle et al. 1993).

Due to the relatively poor spatial resolution of EEGs, it can be difficult to discern between movements of small, specific body parts such as individual fingers. Additionally, EEG results for the same motion can vary drastically between different test subjects and can even vary for the same subject at different times. In spite of this, having the ability to accurately discern between different finger movements is critical in the future of making more functional prostheses. This paper aims to improve upon current models used for EEG classification using transformers and subject transfer, which has not been used for the task of finger classification.

2 Related Works

In 2017, Google Brain and Google Research, teams of top machine learning specialists at Google, published their paper ”Attention is All you Need.” This influential paper introduced a novel neural network architecture known as a Transformer (Vaswani et al. 2017). Transformers employ ”self-attention” which allows the network to identify the importance of elements in the input in relation to other elements in the input.

Transformers are the new preferred architecture for Natural Language Processing (NLP) tasks due to their ability to learn how words in a sentence relate to other words. Furthermore, Transformers are rapidly becoming one of the preferred methods of classification in Computer Vision (CV) due to their ability to extract information about how certain pixels of an image relate to other pixels. This could prove useful in EEG and ECoG classification because regions in those signals are not independent of other regions.

Convolutional Neural Networks (CNNs) are similar to traditional dense artificial neural networks in their functionality. Both serve to extract deep features in a data set to classify new data, but due to the structure of CNNs, CNNs tend to use significantly fewer parameters while maintaining comparable accuracy with two- and three-dimensional data such as images. Instead of connections between every individual node, CNNs utilize ”kernels.” A kernel can be thought of as a small filter that passes over the image and accentuates desirable features.

CNNs excel in mapping convoluted patterns in images to an output variable. They are currently one of the default solutions to prediction problems involving image data as inputs. Consequently, CNNs are a method of identifying deep features of EEG data and classifying them as movement of a specific body part (Sadiq et al. 2022).

In 2021, Ieracetano et al. (2021) showed strong results in using Convolutional Neural Networks (CNNs) to discriminate between open-hand/rest and closed-hand/rest positions using EEG signals. Their work was able to classify pre-hand closed versus rest and pre-hand open versus rest with an accuracy of approximately 90%. Furthermore, they were able to discriminate between preparation of different sub-movements with a precision of approximately 62%.

Kim et al. (2021) discovered that employing Sequential Transfer Learning has the potential to further increase classification accuracy of Motor Imagery (MI) of hand, foot, and tongue movements. Their pre-trained model correctly identified the body part being moved with an accuracy of 63.8% (with transfer learning) compared to a baseline of 61.6% (without transfer learning). Consequently, the performance of their MI-BCI showed improvement when compared to previous CNN-based approaches. In cases with smaller datasets, it can be beneficial to train on similar data prior to fine-tuning on the intended dataset.

In their 2022 paper, Khademi et al. (2022) constructed a hybrid CNN and Long-Short Term Memory (LSTM) deep learning model to classify motor imagery signals from the BCI Competition IV Dataset 2a, a dataset with 4 motor imagery classes (left hand, right hand, feet, and tongue). They achieved a classification accuracy of approximately 90% with their hybrid model of ResNet-50 (a pretrained CNN) and an LSTM.

The electrodes in an Electrocorticogram, or ECoG, are placed directly on the cerebral cortex as opposed to electrodes in an EEG which are placed on the scalp. Consequently ECoG data can be more reliable for predicting fine motor movement due to its improved spatial and temporal resolution as compared to EEG data. As a result, research in 2018 by (Xie, Schwartz, and Prasad 2018) showed promise in being able to distinguish finger movement using a Long-Short Term Memory (LSTM) architecture.

3 Methods

A combination of pre-existing pre-processing methods, ensemble learning, and transfer learning were employed to improve accuracy of classification.

3.1 Obtaining Data

General Overview: Data used was obtained from a large, publicly available electroencephalogram motor imagery dataset for brain-computer interfaces. This dataset contains approximately 60 hours of EEG recordings spread over 75 recording sessions of 13 participants. It contains more than 60000 examples of motor imageries in 4 interaction paradigms. This dataset is one of the largest publicly-available EEG BCI datasets currently published (Kaya et al. 2018). For more details about this dataset, please see Table 1.

Stimuli and Experimental Design: Participants in the 5F (five finger) experiment were first seated in front of a screen. The experiment proceeded as follows: “At the beginning of each trial, an action signal appeared (represented by a number from 1 to 5) directly above the finger whose movement imagery was to be implemented. The action signal remained on for 1s, during which time the participants implemented the corresponding imagery once. The imageries were invoked as a flexion of the corresponding fingers up or down, per the preference of the participant. There was no passive state in this paradigm – each action signal required a response. Single imagery was implemented per action signal. After executing the imagery, participants remained passive until the next action signal presentation.” (Kaya et al. 2018)

3.2 Preprocessing

Overview: Preprocessing was conducted using MATLAB’s PREP Pipeline for standardized EEG preprocessing (Bigdely-Shamlo et al. 2015). The PREP Pipeline allows for automatic selection and rejection of epochs, automatic rejection of channels, removal of line noise, and detrending of the data. Beyond this, electrodes CZ, C3, C4, T3, T4, FZ, F3, and F4 are being utilized due to their close proximity to the motor cortex, the area in the brain that primarily controls motor movement.

Process: All EEG data was imported directly into MATLAB. Channels CZ, C3, C4, T3, T4, FZ, F3, and F4 were extracted from the dataset prior to preprocessing in order to reduce computational demand. These channels were selected due to their proximity to the motor cortex, which is the region of the brain controlling motor function. Data was detrended and line noise was reduced using the standard PREP Pipeline settings. Finally, data was referenced using a RANSAC approach. Data was saved as text files and then converted to MATLAB files to be transformed via Short-time Fourier transform.

Dataset	Type of Data	Classes	Sample Rate (HZ)	Channels	Subjects	Epoch Length	Task
BCI 5F Dataset	EEG	5	200 and 1000	22	8	1 second	Finger Movement

Table 1: Dataset Information

3.3 Overlapping Methods

One preliminary issue that reduced classification accuracy was a lack of data. There were approximately 130 examples of each motor task per subject which is not enough to effectively train a vision transformer. In order to improve classification accuracy, the number of images being fed into the vision transformer was increased. To do this, a process known as ”overlapping” was performed in which multiple offset windows are taken from each epoch.

Standard Overlapping: In standard overlapping, extracted windows are overlapped by a set amount across the full epoch. The typical overlapping percentage ranges from 10% to 90%. For the purpose of this paper, overlapping percentages of 25%, 50%, 75%, and 90% were tested with a window of 0.5 seconds. 75% overlapping showed the most promising results compared to the computational demand and was thus used for further testing.

Triangular Overlapping: Triangular overlapping is an overlapping method proposed by (Atyabi, Fitzgibbon, and Powers 2012) in which desirable sections of the EEG data are overlapped at a higher percentage than other, less desirable sections. For example, in a 2-second long epoch, the beginning of the signal is not useful due to the participants reaction time and the end of the signal isn’t as useful due to mental fatigue. Consequently, overlapping the center of the signal more densely will lead to the model being trained on higher-quality data. A modified triangular overlapping approach was taken to ensure that there are no duplicate samples which would lead to challenges in test/train/validation division prior to training. Unfortunately, this method was infeasible for application to all subjects due to its computational demand and the length of this program.

3.4 Short-Time Fourier Transform

Once the Electroencephalogram data is preprocessed, it would be ready to be classified if the intended classification model was built to handle sequential data, such as a Long Short-Term Memory model (Wang et al. 2018) or a Recurrent Neural Network (Ma et al. 2018). However, sequential data is not useful for a vision-based classifier such as a CNN or a Vision Transformer. Consequently, the data was passed through a script that applied a Short-time Fourier transform (STFT) to each individual channel of every epoch of every trial. STFT converts a function into a spectrogram by determining the sinusoidal frequency and phase content of the function over time. In an STFT, a short window function is slid across the signal over the time axis, and the resulting image is a construction of the individual, short-time dot products of the window function and the signal. An STFT-generated spectrogram of one epoch of EEG data can be seen in Figure 2.

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) = \int_{-\infty}^{\infty} x(t)\omega(t-\tau)e^{-i\omega t}dt$$

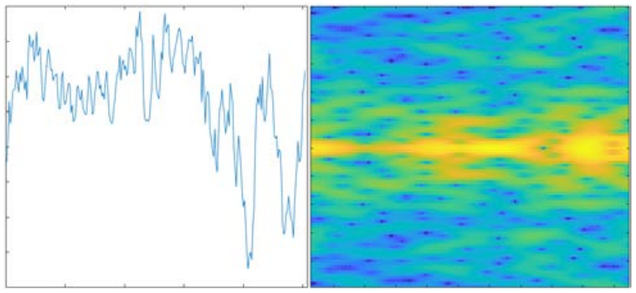


Figure 2: Comparison of raw signal (left) and transformed spectrogram (right).

3.5 Ensemble Learning

Another remedy to the lack of data was the utilization of more channels of the EEG. To train the model on multiple channels, ensemble learning was implemented. Ensemble learning requires simultaneously training of multiple models. Each model was responsible for the classification of a specific channel. To decide the classification of an input, each model in the ensemble ”voted” for the class that it decided the input falls into. This method improves classification accuracy by spreading the data amongst multiple models, reducing the required information storage of each individual model. The drawback of this approach is that it takes significantly longer to train than a single-transformer model. A visualization of this model can be seen in Figure 3.

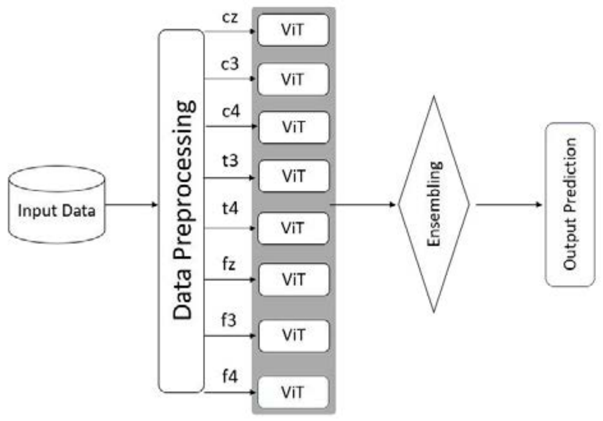


Figure 3: Visualization of Transformer Ensemble

3.6 Implementing a Transformer

The transformers used to classify the electroencephalogram data are BERT-like transformer encoder models that have been pretrained on ImageNet-21k and ImageNet 2012. Pretrained models generally require less data to train and achieve solid performance in less time than comparable untrained models (Kim et al. 2021). Training with limited data is critical in this case as access to large, publicly available EEG datasets is limited.

3.7 Subject Transfer

Transfer learning is the act of using a model trained on one task as a baseline for another task. In this case, the model was trained on seven of the eight subjects before being used to classify the final subject. Two methods of subject transfer were implemented. The first method involved training the transformer ensemble on seven subjects and testing on the last subject with no fine-tuning. The other method involved training the ensemble on seven subjects followed by fine tuning on a randomly selected half of the samples from the final subject.

4 Current Work: Results

4.1 Single Subject: No Fine Tuning

Subject C, Trial 1. Results were obtained by training a separate transformer on channels cz, c3, c4, t3, t4, fz, f3, and f4 of subjects A, B, E, F, G, H, and I before testing on subject C without any retraining. Training was performed with a batch size of 32, a learning rate of 2E-5, and 10 epochs. During testing, each transformer predicted the class based on the input channel. Class was chosen via majority voting. Ties were broken randomly. For these results, see Table 2. Precision, recall, and F1 score are an average across all classes. For specific precision, recall, and F1 scores by class, see the appendix.

Subjects Pretrained On	Precision	Recall	F1
A	NaN	NaN	NaN
A, B	NaN	NaN	NaN
A, B, E	NaN	NaN	NaN
A, B, E, F	NaN	NaN	NaN
A, B, E, F, G	NaN	NaN	NaN
A, B, E, F, G, H	NaN	NaN	NaN
A, B, E, F, G, H, I	NaN	NaN	NaN

Table 2: Single Subject with No Fine Tuning: Results

4.2 Single Subject: With Fine Tuning

Subject C, Trial 1. Results were obtained by training a separate transformer on channels cz, c3, c4, t3, t4, fz, f3, and f4 of subjects A, B, E, F, G, H, and I. This training was followed by randomly splitting Subject C, Trial 1 into 50% train, 50% test. Training was performed with a batch size of 32, a learning rate of 2E-5, and 10 epochs for both pretraining and fine-tuning. During testing, each transformer predicted the class based on the input channel. Class was chosen via majority voting. Ties were broken randomly. For these results, see Table 3. Precision, recall, and F1 score are an average across all classes. For specific precision, recall, and F1 scores by class, see the appendix.

Subjects Pretrained On	Precision	Recall	F1
A	NaN	NaN	NaN
A, B	NaN	NaN	NaN
A, B, E	NaN	NaN	NaN
A, B, E, F	NaN	NaN	NaN
A, B, E, F, G	NaN	NaN	NaN
A, B, E, F, G, H	NaN	NaN	NaN
A, B, E, F, G, H, I	NaN	NaN	NaN

Table 3: Single Subject with Fine Tuning: Results

5 Future Work

5.1 Weighted Voting

In the current transformer ensemble, each transformer gets a ”vote” on which class it believes the input falls into. All votes are weighted equally and the class that gets the majority of votes is the class that the ensemble assigns the input. In reality, some transformers classify their channel more effectively than other transformers in the ensemble. Consequently, weighting votes in favor of more accurate transformers should lead to increased classification accuracy.

5.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been the state-of-the-art for image classification for decades. Consequently, over 70% of deep-learning approaches to EEG classification employ CNNs (Al-Saegh, Dawwd, and Abdul-Jabbar 2021). In the future, a CNN-based classification model will be implemented to compare results with the more recently introduced transformer-based classification model.

6 Conclusion

In order to create more capable prostheses, it is paramount that more work is done to be able to accurately discriminate between small, precise motor movements. As of now, there is much improvement to be done on current models that distinguish between individual finger movements. The goal of this project is to build upon current models and strive toward more accurate CNN- and Transformer-Based EEG and ECoG classification in order to further develop the field of prosthetics.

7 Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Al-Saegh, A.; Dawwd, S. A.; and Abdul-Jabbar, J. M. 2021. Deep learning for motor imagery eeg-based classification: A review. *Biomedical Signal Processing and Control* 63:102172.

Atyabi, A.; Fitzgibbon, S.; and Powers, D. 2012. Multiplication of eeg samples through replicating, biasing, and overlapping. volume 7670, 1–13.

Bigdely-Shamlo, N.; Mullen, T.; Kothe, C.; Su, K.-M.; and Robbins, K. A. 2015. The prep pipeline: standardized preprocessing for large-scale eeg analysis. *Frontiers in neuroinformatics* 9:16.

Ieracitano, C.; Mammone, N.; Hussain, A.; and Morabito, F. C. 2021. A novel explainable machine learning approach for eeg-based brain-computer interface systems. *Neural Computing and Applications* 1–14.

Kaya, M.; Binli, M. K.; Ozbay, E.; Yanar, H.; and Mishchenko, Y. 2018. A large electroencephalographic motor imagery dataset for electroencephalographic brain computer interfaces. *Scientific data* 5(1):1–16.

Kim, D.-K.; Kim, Y.-T.; Jung, H.-R.; Kim, H.; and Kim, D.-J. 2021. Sequential transfer learning via segment after cue enhances the motor imagery-based brain-computer interface. In *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, 1–5.

Ma, X.; Qiu, S.; Du, C.; Xing, J.; and He, H. 2018. Improving eeg-based motor imagery classification via spatial and temporal recurrent neural networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 1903–1906. IEEE.

Sadiq, M. T.; Aziz, M. Z.; Almogren, A.; Yousaf, A.; Siuly, S.; and Rehman, A. U. 2022. Exploiting pretrained cnn models for the development of an eeg-based robust bci framework. *Computers in Biology and Medicine* 105242.

Towle, V. L.; Bolaños, J.; Suarez, D.; Tan, K.; Grzeszczuk, R.; Levin, D. N.; Cakmur, R.; Frank, S. A.; and Spire, J.-P. 1993. The spatial location of eeg electrodes: locating the best-fitting sphere relative to cortical anatomy. *Electroencephalography and clinical neurophysiology* 86(1):1–6.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems* 30.

Wang, P.; Jiang, A.; Liu, X.; Shang, J.; and Zhang, L. 2018. Lstm-based eeg classification in motor imagery tasks. *IEEE transactions on neural systems and rehabilitation engineering* 26(11):2086–2095.

Xie, Z.; Schwartz, O.; and Prasad, A. 2018. Decoding of finger trajectory from ecog using deep learning. *Journal of neural engineering* 15(3):036009.

8 Appendix

No Fine-Tuning	**A**					**A,B**					**A,B,E**				
Channel	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Precision	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
Recall	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
F1 Score	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

No Fine-Tuning	**A,B,E,F**					**A,B,E,F,G**					**A,B,E,F,G,H**				
Channel	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Precision	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
Recall	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
F1 Score	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

No Fine-Tuning	**A,B,E,F,G,H,I**				
Channel	1	2	3	4	5
Precision	.000	.000	.000	.000	.000
Recall	.000	.000	.000	.000	.000
F1 Score	.000	.000	.000	.000	.000

With Fine-Tuning	**A,B,E,F,G,H,I,C**				
Channel	1	2	3	4	5
Precision	.000	.000	.000	.000	.000
Recall	.000	.000	.000	.000	.000
F1 Score	.000	.000	.000	.000	.000

3DChromoTwist: Development of a 3D Chromosome Structure Reconstruction Game for Educational Purposes

Marcin Pawlukiewicz
University of Rhode Island
45 Upper College Rd
Kingston, RI 02881
mapawlukiewicz@uri.edu

Oluwatosin Oluwadare
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
ooluwada@uccs.edu

Abstract

The 3D structure reconstruction of the chromosome is important so that a better knowledge of chromosome activity can aid us in understanding DNA replication, gene regulation, genome interaction, genome folding, and genome function. The High-throughput Chromosome Conformation Capture (Hi-C) technique incorporates chromosome conformation capture approach with the power of the Next Generation Sequencing technologies to study the 3D chromatin organization. From the Hi-C data, we can infer the contact or interaction frequency (IF) matrix which describes the level of interaction between the chromosome bins. In this work, we propose the development and design of a single-player bioinformatics game called 3DChromoTwist. The project's objective is to interest non-scientists in learning about three-dimensional (3D) chromosomal structure. A reduced Hi-C contact matrix representing the final, complete structure is provided to players along with a 3D fragment of a chromosome. Players must then solve the puzzle by moving gene loci of 3D chromosomal structures until they form the desired relationships between the folds of the chromosome. We expect that 3DChromoTwist will positively influence the progression of science by introducing a new set of learners to the world of genetics. Ultimately, this will increase public scientific literacy and public engagement with science and technology.

Introduction

High throughput chromosome conformation capture (Hi-C) is a technique for evaluating chromatin's spatial arrangement in a cell. The technique counts how many interactions there are between genome loci. The genome loci are detectable in a 3D structure when they are close together yet separated by many nucleotides in a linear genome. (Hakim and Misteli 2012).

The process involves crosslinking DNA such that adjoining areas are linked together. The regions can then be quickly identified. When cell genomes are cross-linked with formaldehyde, chromatin crosslinking begins. DNA fragments are then cut with a restriction enzyme, fragments are sealed with a biotin marker, and ligated. Chimeric DNA fragments are then created using reversed crosslinking. After DNA purification, streptavidin is used to pull down the DNA, after which the fragments are sequenced. Hi-C sequencing will generate sequenced pair reads that identify their nucleotides. These are then mapped to a reference genome and filtered for noise, ultimately preprocessed to produce an IF matrix (Sati and Cavalli 2017). Hi-C sequencing results in a comparison of multiple DNA fragments to each other. As a result, it scans the locus of the close pairs and maps their relations onto a N*N matrix, where N is the number of chromosome fragments. In a Hi-C experiment, each element in the matrix comprises a count of reading pairs that connect two homologous chromosome regions. As a result, the chromosomal contact matrix is symmetric and represents all observable interactions between the regions of a chromosome (Lieberman-Aiden et al. 2009). This data can be used in the 3D reconstruction of chromosomes. The reconstruction could be used to study DNA replication, gene regulation, genome interaction, genome folding, and genome function (Oluwadare, Zhang, and Cheng 2018). We can see that the number of relationships between chromosome regions is proportional to the distance in the 3D structure, therefore we can make predictions relying on that information (Lieberman-Aiden et al. 2009). Over the years, many 3D chromosome reconstruction algorithms have been developed, (Oluwadare, Highsmith, and Cheng 2019; MacKay and Kusalik 2020) give a comprehensive review of these methods and their strengths.

The way chromosomes fold helps us to understand data about the intricate connection between chromatin structure, quality movement, and the useful condition of the cell (Woodcock and Ghosh 2010). The genome must be packed properly into the nucleus. Otherwise, serious diseases, such as congenital malformations (ex. fewer or too many fingers) or cancer, may occur (Gorkin 2017). Knowledge of how genes act can help with the reconstruction of the entire genetic or biochemical pathways. This is essential to our understanding of metabolism, signal transduction, and other developmental or psychological processes (Dekker et al. 2002). Folding DNA into chromosomes is crucial as this is something that makes us, us. All the processes in our cell must occur correctly. This prevents tangling and damage during cell division. In the case of pregnancy, if the embryo misses a chromosome, it means that there is some genetic damage, and this can prevent a successful pregnancy (Xie et al. 2021). Folding DNA into chromosomes makes the genetic material fit inside a cell but allows for distant interactions between them. This process is important as this regulates gene activities (Banigan et al. 2020).

Chromosomes consist of tightly packed DNA, wound up in order to condense a vast amount of genetic information into a smaller volume. DNA consists of a double helix, which is then wrapped around histone proteins to create nucleosomes. From there, nucleosomes are coiled into chromatin fiber, then looped to condense the chromatin into the final form of a fully assembled chromosome. Our game takes place at the point nucleosome creation, the point at when the relationships between segments of DNA can be analyzed and used to predict what the final chromosomal structure will be. This is what is calculated then given to solve to players via chromosome segments based on the Hi-C matrices (Alberts et al. 2002).

Chromosomes are part of every nucleus in every cell of our body. Visualization of the folding process is crucial for deeper understanding of the above addressed problems. The game helps not only to understand the processes in our body but also teaches the fundamental knowledge necessary in every biological field. Thus, the game influences the progression of science by equipping non-experts with biological knowledge, engaging them through game-play to contribute to the world of genetics.

The very first game was presented over fifty years ago (Ford 2012). Since then, many new games have appeared. Today, we can count them in millions. Both scientific and non-scientific games played have the element of entertainment. Without this, players will not be interested in the game. Moreover, the game should also be informative by presenting the player with some useful knowledge. Thus, making a game that is simultaneously fun and educational is important.

We believe that 3D structures are an informative and easily understood way to appeal to non-scientific audiences by giving reasonably accurate visualizations. We use current methods and algorithms to make the game as much realistic as possible to enable the possibility for new learners to make new discoveries and contribute to the scientific world.

This paper describes 3DChromoTwist, a game created to educate about chromosomal structure through the use of Hi-C data. The game is directed toward a broad audience and thus our development is adjusted as we expect people from non-scientific backgrounds to participate. The design and easy level are simplistic and, as we proceed to the hard level, more complex. The user can start from any level and move to more advanced levels at later stages as appropriate skills have been developed. Both design and movement of the structures need not be over complicated to reach the desired audience.

Related Work

Our inspiration is the popular online multiplayer game “Fold it” (Cooper et al. 2010). The game follows an educational approach and falls in the field of biology. In the game, players learn how protein folding mechanisms work. The design is simplified for non-expert players however, the game has many features. We noticed that the game garnered a lot of popularity over the years and many papers have been published regarding it. As we noticed the huge positive impact of this game, we decided to implement similar solutions into chromosomal structures.

Our work differs by applying the concepts of structure formation to 3D chromosome structure. As opposed to folding parts of a protein to prevent viral attacks within the structure of the protein, chromosome segments are moved around to reconstruct the natural positioning of the chromosomal components into their correct place, allowing the complete chromosome structure and shape to be revealed to the player once a level is successfully solved. This also played using a given Hi-C matrix, a unique component when compared to “Foldit.” Moreover, there is no current related work in chromosome structure prediction. Thus, this will pioneer a new research direction towards this goal.

Implementation

Table 1: Key description

Key	Description
H	Hide/ Show the menu.
L	Surrender.
O	Hide/ Show the outline of the objects.
Q	Reset the bin choice.
R	Hide/ Show the scoreboard (top 5 players).
+/=	Choose bin with index higher by one.
-	Choose bin with index lower by one.
0	Choose a bin with index increased by 10.
1-9	Pick index from 1 to 9 in the current teens.
Scroll	Zoom in and out.
ESC	Quit the game.

The game has been created as a 3D structure where the user can freely move around the objects to obtain a better understanding of the structure with the use of the following keys: W, S, A, and D. While camera motion is controlled with the use of the mouse, holding the keys moves the player

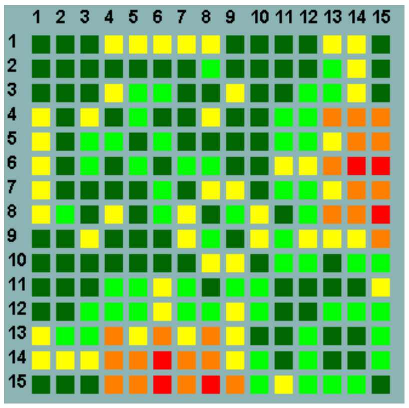
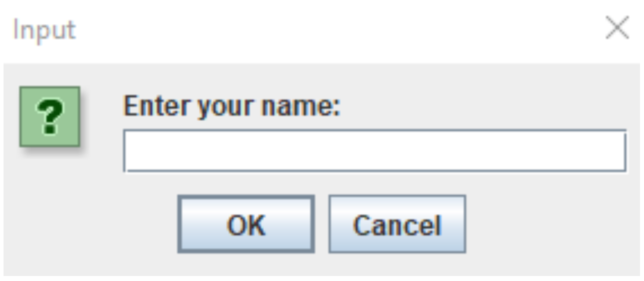


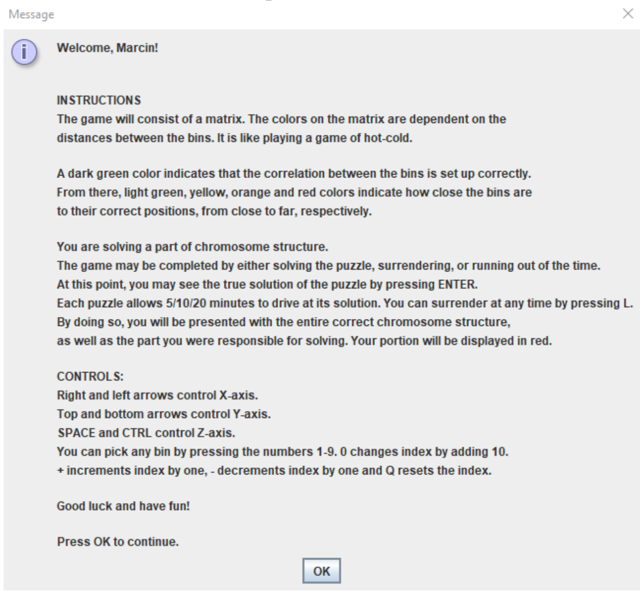
Figure 1: Simplified Hi-C matrix in the game, level normal.



Input

Enter your name:

(a) Prompt to enter the name.



Message

Welcome, Marcin!

INSTRUCTIONS
The game will consist of a matrix. The colors on the matrix are dependent on the
distances between the bins. It is like playing a game of hot-cold.

A dark green color indicates that the correlation between the bins is set up correctly.
From there, light green, yellow, orange and red colors indicate how close the bins are
to their correct positions, from close to far, respectively.

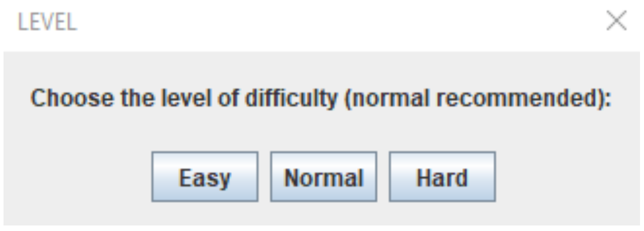
You are solving a part of chromosome structure.
The game may be completed by either solving the puzzle, surrendering, or running out of the time.
At this point, you may see the true solution of the puzzle by pressing ENTER.
Each puzzle allows 5/10/20 minutes to drive at its solution. You can surrender at any time by pressing L.
By doing so, you will be presented with the entire correct chromosome structure,
as well as the part you were responsible for solving. Your portion will be displayed in red.

CONTROLS:
Right and left arrows control X-axis.
Top and bottom arrows control Y-axis.
SPACE and CTRL control Z-axis.
You can pick any bin by pressing the numbers 1-9. 0 changes index by adding 10.
+ increments index by one, - decrements index by one and Q resets the index.

Good luck and have fun!

Press OK to continue.

(b) Instructions of the game.



LEVEL

Choose the level of difficulty (normal recommended):

(c) Level of difficulty.

Figure 2: The beginning of the game.

forward, backward, left, and right, respectively. When starting the game, the player is first asked to enter a name 2a. Once okay is clicked, the player is then presented with the instructional prompt 2b. After acknowledging the instructions, the player needs to choose a level of difficulty as presented in figure 2c. Next, on the display, the player can see the following additional key options described in table 1.

Apart from all this, the player can see a simplified version of the Hi-C data matrix. The data for the chromosome structure has been used from the following paper, "HSA: integrating multi-track Hi-C data for genome-scale reconstruction of 3D chromatin structure" (Zou, Zhang, and Ouyang 2016). We took a hundred points x, y, and z that are loaded onto the

Table 2: Distance description

Name	Distance	Color
Completed	The correct position of the object with allowed margin of error up to 2 units.	Dark green
Very Close	Between 2 and 4 units.	Green
Close	Between 4 and 8 units.	Yellow
Far	Between 8 and 15 units.	Orange
Very Far	Above 15 units.	Red

The distance is measured by the Euclidean distance formula. Every center of the loci (x, y, z) is compared with every other object.

level as every game begins. Next, depending on the choice of difficulty, the player receives a random consecutive number of bins that are taken from the loaded data. Each bin is generated in a randomized position. To ease the difficulty of the game, the x coordinates have been randomized with some precision to the original spot. The player can see the Hi-C matrix with the colors (figure 1) defined as presented in table 2. This is an estimation that shows the distance between the current location and the correct final position of the bins. Using Euclidean distance, the program checks the correctness of the correlation between the bins as the player progresses through the level. The colors on the matrix progress towards red as the position of the bins moves away from the correct position.

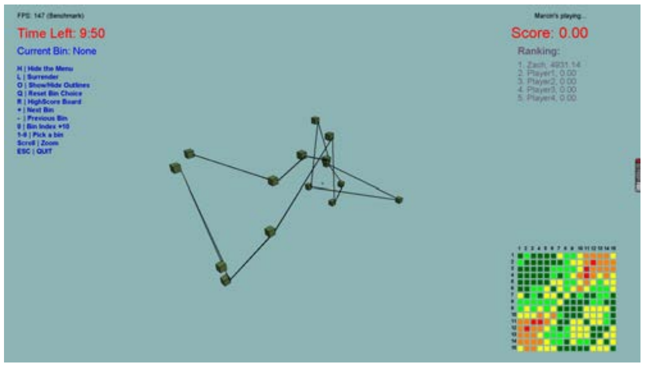


Figure 3: Start of the game, level normal.

The player can pick a bin using the keyboard. The numbers 1 to 9 can pick a bin from the current teens, while 0 increases the bin by 10. The following keys "-" and "=" decrease and increase the current bin by one accordingly. The active bin can be noticed by a color change to red (figure 4). Although the bins have been indexed properly with the order of the x-coordinate, the player needs to figure out the index on its own while playing.

There are two important components in the game that are responsible for motivation and engagement. The score is the first factor. The player gains points as the matrix becomes better and there is a less red and more dark green color on it. The starting matrix is set up as a score of zero. Any deterioration of the matrix will encourage the player make up for this if the player wants to gain points. The scoring algorithm

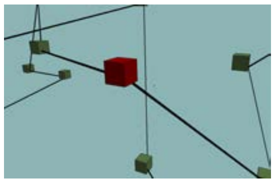


Figure 4: Active bin marked on red.

has been adjusted correspondingly, so no matter which matrix the user starts out with, it still achieves the same number of points. Another component is time. The player has a limited amount of time to complete the game. Depending on the level the player chooses, the time increases with difficulty. If the player does not make it on time, the game will end (figure 5). The score will be saved and displayed along with a "Game Over" message. However, if the player is able to complete the structure before time runs out, the score will be increased with bonus points. Depending on the time, the score will be adjusted accordingly. The player will be prompted with a "Victory" message. The player also has the option to surrender. In this instance, there are no bonus points added and the current score will be saved. To complete the level, the player needs to reposition loci according to clues received from the simplified Hi-C data matrix. The matrix is interactive and changes colors as the player changes distance. As loci get closer to their proper destination, the color changes to dark green. However, as the player gets further from the proper final destination, the color changes to red. Once the matrix is fully dark green, the level is complete (figure 6).

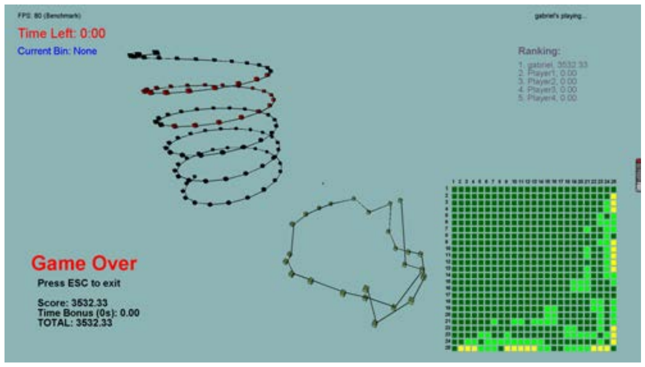

FPS: 80 (Benchmark)
Time Left: 0:00
Current Bin: None
gabriel's playing...
Ranking:
1. gabriel, 3532.33
2. Player1, 0.00
3. Player2, 0.00
4. Player3, 0.00
5. Player4, 0.00
Game Over
Press ESC to exit
Score: 3532.33
Time Bonus (0s): 0.00
TOTAL: 3532.33

Figure 5: End of the game with the solution, level hard.

To make the game more competitive, each level has its own high score board. The top 5 players are always displayed with their scores. The player can show and hide the scoreboard at any point during the game. Once the game is completed, the scoreboard is displayed. Each level of difficulty has its own scoreboard. All the scores are saved into a text file that is secured by a number generated depending on the strings, results, and characters. Therefore, any attempts to change the records manually will result in resetting the file. Similarly, a change in coordinates in the text file will have similar results.

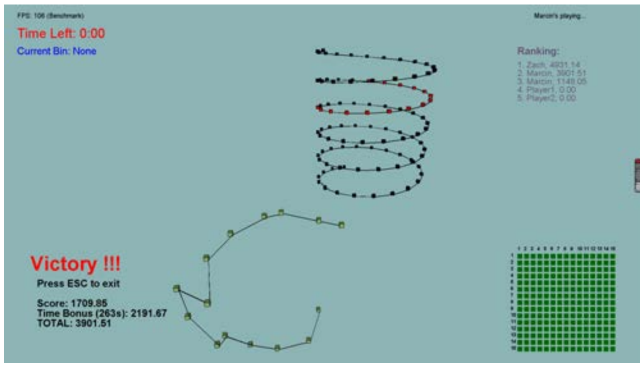

FPS: 106 (Benchmark)
Time Left: 0:00
Current Bin: None
Marcin's playing...
Ranking:
1. Zach, 4931.14
2. Marcin, 3901.51
3. Marcin, 1148.05
4. Player1, 0.00
5. Player2, 0.00
Victory !!!
Press ESC to exit
Score: 1709.85
Time Bonus (263s): 2191.67
TOTAL: 3901.51

Figure 6: End of the game with the solution, level normal.

Upon completion of the game, the player, besides the "Game Over" or "Victory" message, will be allowed to see the solution by pressing "ENTER." Once the key is pressed, the player will see the level's complete structure as well as the solution next to it (figure 6). The solution represents the entire chromosome structure built out of 100 loci. However, because the player was required only to play with between 7 and 25 loci, all the loci used in the current game and chosen by the randomizing algorithm will be highlighted in red. The solution post-view is the last step in the game (figure 7). The player can choose to close the game by pressing "ESC" on the keyboard and to start the game again, the player is required to rerun the jar file.

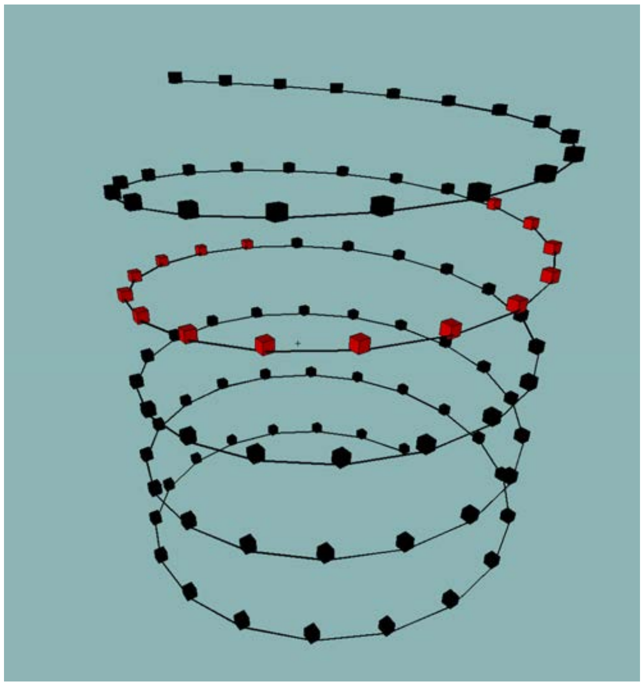


Figure 7: Chromosome structure.

We expect that the players, with the help of the given solutions in the form of the simplified Hi-C matrices, will be able to solve the puzzle. Naturally, replication of the same chromosome is extremely difficult, as there are many possibilities involved in its 3D structure. Hence, the acceptable margin of error is equal to 2 units.

Conclusion

The 3DChromoTwist game is playable by anyone using any type of personal computer thanks to the design and simplicity of the 3D software used to develop it. The fact that the results of the 3D chromosomal structure prediction are unknown posed the biggest design challenge. Even the most sophisticated structures offered by the game are uncertain, but the game nevertheless directs the player to them.

The game is very intuitive from the very first level to make it accessible as the player progresses. By making 3DChromoTwist available to users, we hope to demonstrate a fresh method for teaching chromosomal structure prediction and foster a deeper comprehension of the topic. Moreover, the game allows the visualization of these structures in 3-Dimension, giving a realistic perspective of the chromosome behavior. The game is a great complement to today's textbooks since it makes it easier to understand and picture the structure in real time.

The goal of the game is to gradually increase the level of difficulty while yet maintaining a pleasurable experience for the player. The players' problem-solving abilities may be enhanced by 3DChromoTwist, which can also assist in the resolution of actual scientific conundrums.

We intend to keep working on 3DChromoTwist. As we believe the game can advance research by introducing a new group of students to the field of genetics, we aim to improve the game design and add new levels throughout time. In the end, this will elevate public interest in science and technology as well as public scientific literacy. If even a fraction of the effort that goes into playing computer games can be directed toward scientific research, we personally believe that scientific advancement is achievable.

Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Alberts, B.; Johnson, A.; Lewis, J.; Raff, M.; Roberts, K.; and Walter, P. 2002. *Molecular biology of the cell*. Garland Science, 4th ed edition.

Banigan, E. J.; van den Berg, A. A.; Brandão, H. B.; Marko, J. F.; and Mirny, L. A. 2020. Chromosome organization by one-sided and two-sided loop extrusion. *eLife* 9:e53558.

Cooper, S.; Treuille, A.; Barbero, J.; Leaver-Fay, A.; Tuite, K.; Khatib, F.; Snyder, A. C.; Beenen, M.; Salesin, D.; Baker, D.; and Popović, Z. 2010. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, 40–47. New York, NY, USA: Association for Computing Machinery.

Dekker, J.; Rippe, K.; Dekker, M.; and Kleckner, N. 2002. Capturing chromosome conformation. *Science* 295(5558):1306–1311.

Ford, W. K. 2012. Copy game for high score: The first video game lawsuit. *Journal of Intellectual Property Law* 20:1–3.

Gorkin, D. 2017. Genomic technologies for studying 3d genome organization. *Encode* 1–12.

Hakim, O., and Misteli, T. 2012. SnapShot: Chromosome conformation capture. *Cell* 148(5):1068–1068.e2.

Lieberman-Aiden, E.; van Berkum, N. L.; Williams, L.; Imakaev, M.; Ragoczy, T.; Telling, A.; Amit, I.; Lajoie, B. R.; Sabo, P. J.; Dorschner, M. O.; Sandstrom, R.; Bernstein, B.; Bender, M. A.; Groudine, M.; Gnirke, A.; Stamatoyannopoulos, J.; Mirny, L. A.; Lander, E. S.; and Dekker, J. 2009. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science* 326(5950):289–293.

MacKay, K., and Kusalik, A. 2020. Computational methods for predicting 3d genomic organization from high-resolution chromosome conformation capture data. *Briefings in Functional Genomics* 19(4):292–308.

Oluwadare, O.; Highsmith, M.; and Cheng, J. 2019. An overview of methods for reconstructing 3-d chromosome and genome structures from hi-c data. *Biological Procedures Online* 21(1):7.

Oluwadare, O.; Zhang, Y.; and Cheng, J. 2018. A maximum likelihood algorithm for reconstructing 3d structures of human chromosomes from chromosomal contact data. *BMC Genomics* 19(1):161.

Sati, S., and Cavalli, G. 2017. Chromosome conformation capture technologies and their impact in understanding genome function. *Chromosoma* 126(1):33–44.

Woodcock, C. L., and Ghosh, R. P. 2010. Chromatin higher-order structure and dynamics. *Cold Spring Harbor Perspectives in Biology* 2(5):a000596–a000596.

Xie, D.; Yang, W.; Fang, J.; Li, H.; Xiong, L.; Kong, F.; Wang, A.; Liu, Z.; and Wang, H. 2021. Chromosomal abnormality: Prevalence, prenatal diagnosis and associated anomalies based on a provincial-wide birth defects monitoring system. *Journal of Obstetrics and Gynaecology Research* 47(3):865–872.

Zou, C.; Zhang, Y.; and Ouyang, Z. 2016. Hsa: Integrating multi-track hi-c data for genome-scale reconstruction of 3d chromatin structure. *Genome Biology* 17(1).

RECSplice: Splice Site Prediction Using Recurrent Neural Networks

Nicole Baugh
North Carolina State University
ncbaugh@ncsu.edu

Oluwatosin Oluwadare
University of Colorado Colorado Springs
ooluwada@uccs.edu

Abstract

Post-transcriptional slicing of mRNA occurs when regions of RNA that do not encode for protein expression (introns) are removed from regions that do (exons). Accurate identification of splice sites in DNA sequences plays an important role in the structural and functional identification of eukaryotic genes. Thus, accurate splice site detection is essential for biological and medical tools of diagnosis and treatment. However, current computational models for splice site prediction are lacking in efficiency and biological prowess. With this in mind, we propose RECSplice, a deep learning Recurrent Neural Network (RNN) architecture for splice site prediction. RNN algorithms imitate the sequential nature of RNA, tracking the location and order of nucleotide sequences to better identify intron and exon genomic patterns with long term dependencies in addition to other important biological characteristics. In this work, we will compare the ability of existing state-of-the-art splice site prediction algorithms to accurately identify splice site locations in *Homo sapiens, Drosophila melanogaster, and Arabidopsis thaliana* with RECSplice.

Introduction

Alternative splicing plays a significant contribution in expanding the protein diversity of humans. Nearly 95% of human genes undergo splicing, where non-coding regions of mRNA (introns) are separated from coding regions of mRNA (exons) through the spliceosome complex (Pan et al. 2008). A splice site is classified as the boundary between an exon and an intron. Post-splicing, the coding regions are sealed back together before undergoing translation for protein expression. However, only 3% of the entire genomic region is composed of coding regions that can reach this point (Goel, Singh, and Aseri 2013). Though the other 97% of the genome is removed prior to protein expression, genome-wide association studies have identified over 6,000 diseases that have contributing mutations or predispositions located in non-coding regions (Hindorff et al. 2009). Thus, identifying the intron regions of DNA can be a powerful tool for diagnosis.

Machine learning algorithms have increasingly been employed to perform human genome sequencing in order to identify these splice site locations. One such architecture, recurrent neural networks (RNN), mimic the sequential nature of genomic sequences. Trained on a data set, RNN's use internal state memory as well as the present input in order to determine an algorithm-based classification (Sherstinsky 2020). As genomic nucleotide sequences are not independent of each other, this internal state memory will allow the deep learning network to take advantage of known biological characteristics of splice site locations. One such characteristic includes the consensus Adenine-Guanine at the intersection of the intron-exon border, or the acceptor site, and the consensus Guanine-Thymine at the intersection of the exon-intron border, or the donor site (Goel, Singh, and Aseri 2015). Accurate prediction of splice sites must include both the canonical AG/GT sites as well as alternate splicing locations that enhance protein diversity by allowing the production of different proteins from a single gene. As AG and GT rich sites exist elsewhere in the mRNA genome, false positives can result from mistakenly classifying these pseudo canonical sites as splice sites (Ruohan et al. 2019).

This study focuses on splice site prediction using a RNN architecture that considers canonical and non-canonical sites in order to increase the efficiency and accuracy of splice site prediction as an important biological and medical tool.

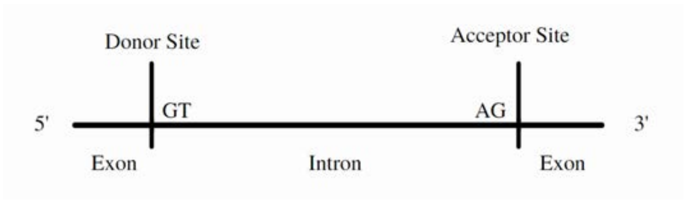


Figure 1: A depiction of acceptor and donor splice sites with canonical sequences.

Related Work

Accurate identification of splice sites plays a central role in understanding the structure of genes in eukaryotes. Non-deep learning and deep learning (DL) algorithms and models are an active research area in Bioinformatics for splice site genomic analysis. Many models struggle with ineffective classification from raw data, model overfitting, and problematic sequence pattern discovery.

One such non-DL method is GeneSplicer, a combination of maximal dependence decomposition decision trees and enhanced Markov models that focuses only on small windows surrounding splice junctions (Pertea, Lin, and Salzberg 2001). GeneSplicer used two second-order Markov models to model the coding and non-coding regions that always surround a splice site to reduce false positives, a known issue in splice site prediction across all models.

Ruohan et al. developed SpliceFinder, a DL model that utilizes a convolutional neural network (CNN), to reduce false positives while maintaining recall for ab initio prediction (Ruohan et al. 2019). The CNN was trained on human genomic data (GRCh38) and achieved a 96.5% accuracy. Another DL technique, DeepSplicer, employed a CNN in tandem with grid search methods to find optimal hyperparameters to enhance classification accuracy of acceptor and donor sites (Akpokiro, Oluwadare, and Kalita 2021). Using five-fold cross-validation, DeepSplicer was able to achieve an accuracy of 96.65%.

The iss-CNN model similarly uses grid search methods to tune hyperparameters for sequence based prediction (Tayara, Tahir, and Chong 2019). After testing four different models with varying convolutional layer parameters, iss-CNN achieved an accuracy of 96.66% on donor sites using two convolution layers. However, iss-CNN was only able to achieve an accuracy of 93.57% on acceptor sites using a single one-dimensional convolution layer.

Zuallaert et al. developed the CNN model SpliceRover with the intention of classifying splice site locations as well as understanding the 'black box' nature of the convolutional architecture, or the relatively little understood learned reasoning mechanisms found within the computational layers (Zuallaert et al. 2018). SpliceRover was trained on only canonical splice site data and used the DeepLIFT (Deep Learning Essential FeaTures) algorithm for evaluation.

Some deep learning algorithms aim to recognize splice sites in primary DNA sequences that have yet to undergo transcription as opposed to traditionally examined mRNA sequences that have yet to undergo translation. One such CNN model, Splice2Deep, trained two independent models, one for acceptor sites and one for donor sites, while using flanking regions and exon periodicity of three features of DNA (Albaradei et al. 2020). Another DNA-focused model, DeepSS, utilized the same CNN architecture in two different applications: DeepSS-C for splice site classification and DeepSS-M for splice site sequence pattern detection (Du et al. 2018). DeepSS-C generated better results on donor splice sites than on acceptor sites, a trend commonly seen across machine learning algorithms for splice site prediction. This trend is largely attributed to a more conservative GT sequence at the donor location than that of the AG sequence at the acceptor location.

Despite the accuracy of CNN models, they are unable to take into account the sequential, nucleotide-dependent nature of RNA, leaving room to accommodate prediction performance enhancements.

RNN's have been employed in epigenome-based splice site prediction in order to account for spatiotemporal patterns found beyond the genomic sequence. Lee et al. developed a RNN prediction model to account for histone modifications and mRNA accessibility, placing splice site prediction in an epigenomic context (Lee et al. 2020). This epigenome based model supports the application of RNN architecture for sequential information as well as time-direction data such as mRNA, which runs along genomic coordinates in the 5' to 3' direction. An RNN Long Short Term Memory (LSTM) model is a promising proposal to account for spatiotemporal patterns while allowing for sufficient training on expansive genomic datasets.

Preliminary Work

Data

To demonstrate RECSplice's adaptability, we trained the model on three datasets from organisms: *Homo sapiens, Drosophila melanogaster, and Arabidopsis thaliana.* The model was trained on a sequence of approximately 10,000 nucleotides long for each dataset. These reference genomic sequence datasets and their corresponding annotation sequences were downloaded from NN269, a database tool that organizes genomic reference data.

Input Data

During preprocessing of the genomic reference sequences, one-hot encoding was performed. One-hot encoding transforms the four nucleotide bases into numerical vectors composed of 0 and 1. For example, Adenine (A) is [1 0 0 0], Cytosine (C) is [0 1 0 0], Guanine (G) is [0 0 1 0], and Thymine (T) is [0 0 0 1]. This results in a matrix that is the length of the sequence data x 4 that serves as the input for the model.

Methods

An RNN is a neural network that is distinguished through its use of internal state memory. Data such as genomic sequences follow a dependent sequential order that RNN's can account for with this memory loop. A known issue with RNN models includes vanishing gradients as the algorithm uses backpropogation through time for training. In order to combat this issue, we propose using a LSTM model, a form of RNN, to predict splice site locations. LSTM's are advantageous in the way that they allow long term dependencies with a stable gradient. The proposed architecture will use three gates within the LSTM, including an input, a forget, and an output gate. The forget gate allows the algorithm to disregard unnecessary states based on a sigmoid function (0 indicating completely disregard, and 1 indicating completely remember) in order to avoid the vanishing gradient problem.

RECSplice's current model uses a two layer LSTM architecture, connected through a dense classification layer. A dropout of 0.3 and categorical cross entropy loss were used. Figure 2 illustrates the LSTM model. Additionally, 5-fold cross validation is used in order to ensure optimal selection of a final architecture for training and testing. A separate model for donor and acceptor splice sites was used.

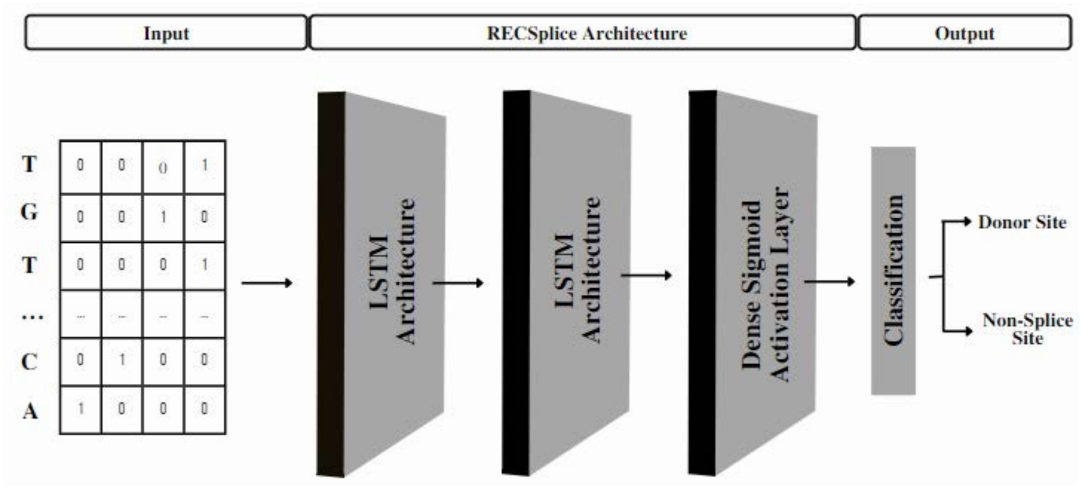


Figure 2: Current RECSplice architecture for donor sites.

Results and Discussion

Validation

Initially, an input length of 400 nucleotides was used, resulting in an accuracy that plateaued at 75%, illustrated in Figure 3. However, other RNN architectures employed in genomic data prediction problems have shown higher accuracy rates when using smaller nucleotide input lengths. For example, Tahir et al. developed an LSTM algorithm for the prediction of N7-Methylguanoisine sites found in RNA using an input length of 41 nucleotides, achieving an accuracy of 95.95% (Tahir et al. 2022). Canatalay and Ucan utilized an input length of 70 nucleotides in their LSTM-RNN and GRU method architecture to achieve an accuracy of 96.1% for exon prediction and splice site mapping (Canatalay and Ucan 2022).

Based on these studies, an input length of 100, 80, 60, and 50 nucleotides were each tested in the initial RECSplice architecture, using 20 epochs with a batch size of 50. The results in Figure 4 were obtained.

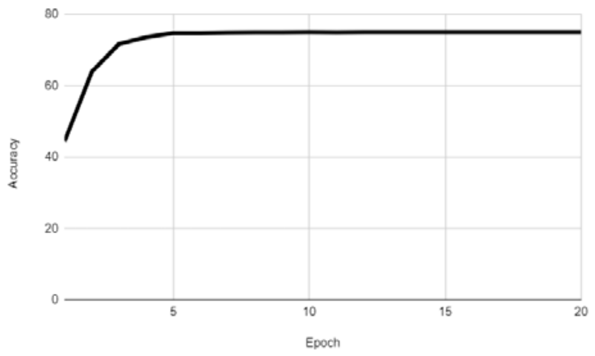


Figure 3: Initial results with nucleotide length of 400.

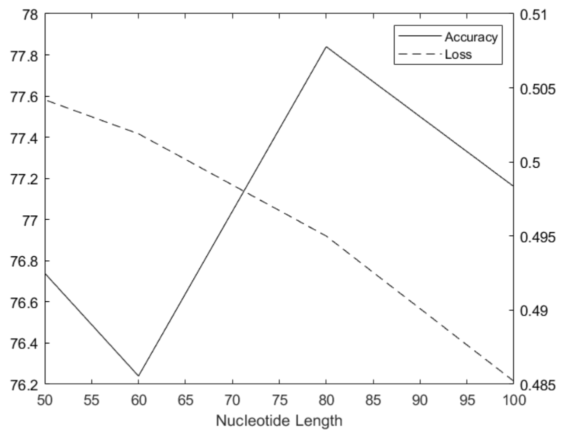


Figure 4: Accuracy and Loss values using differing nucleotide input lengths.

Comparative Metrics

Independent models for classification of donor and acceptor splice site locations were created using 5-fold cross validation and a 70/30 percent dataset split. A sequence length of 80 was maintained as input based on Figure 4.

In evaluating RECSplice's performance in predicting splice sites, accuracy was used as a comparison metric. Precision and recall metrics will be added to future RECSplice models.

Accuracy is given by:

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

The preliminary results presented in this section focus only on donor sites, though acceptor locations will be used

Table 1: Comparing RECSplice and State-of-the-Art CNN splice site prediction models accuracy on donor locations.

Organism	RECSplice	DeepSplicer	SpliceFinder	DeepSS	Splice Rover
Homo Sapiens	78.84	93.13	92.05	94.27	95.49
Drosophila Melanogaster	76.67	90.03	N/A	N/A	89.85
Arabidopsis Thaliana	75.42	89.78	88.96	N/A	92.25

in future models. Table 1 shows the accuracy comparison of RECSplice to State-of-the-art models. Currently, the RECSplice model is not competitive with the current state-of-the-art models for splice site classification. Architectural and hyperparameter improvements will be made in order to become competitive. A deep bidirectional LSTM architecture is currently being worked on in order to improve the RECSplice model.

Conclusion

Splice site prediction modeling is in need of an efficient and cost effective technique that allows for accurate classification of splice site locations. If the final RNN algorithm in this paper proves more accurate than current state-of-the-art CNN splice site prediction algorithms, it would prove a critical development for medical diagnosis and treatment.

Acknowledgement

The work reported here is supported by the NSF REU funding towards the University of Colorado Colorado Springs and Oluwatosin Oluwadare.

References

Akpokiro, V.; Oluwadare, O.; and Kalita, J. 2021. DeepSplicer: An Improved Method of Splice Sites Prediction Using Deep Learning. *IEEE International Conference on Machine Learning and Applications*.

Albaradei, S.; Magana-Mora, A.; Thafar, M.; Uludag, M.; Bajic, V. B.; Gojobori, T.; Essack, M.; and Jankovic, B. R. 2020. Splice2Deep: An Ensemble of Deep Convolutional Neural Networks for Improved Splice Site Prediction in Genomic DNA. *Gene*.

Canatalay, P. J., and Ucan, O. N. 2022. A Bidirectional LSTM-RNN and GRU Method to Exon Prediction Using Splice-Site Mapping. *MDPI*.

Du, X.; Yao, Y.; Diao, Y.; Zhu, H.; Zhang, Y.; and Li, S. 2018. DeepSS: Exploring Splice Site Motif Through Convolutional Neural Network Directly From DNA Sequence. *IEEE Access*.

Goel, N.; Singh, S.; and Aseri, T. C. 2013. A Review of Soft Computing Techniques for Gene Prediction. *ISRN Genomics*.

Goel, N.; Singh, S.; and Aseri, T. C. 2015. An Improved method for Splice Site Prediction in DNA Sequences Using Support Vector Machines. *Procedia Computer Science*.

Hindorff, L. A.; Sethupathy, P.; Junkins, H. A.; Ramos, E. M.; Mehta, J. P.; Collins, F. S.; and Manolio, T. A. 2009. Potential Etiologic and Functional Implications of Genome-Wide Association Loci for Human Diseases and Traits. *Proceedings of the National Academy of Sciences*.

Lee, D.; Zhang, J.; Liu, J.; and Gerstein, M. 2020. Epigenome-Based Splicing Prediction Using a Recurrent Neural Network. *PLoS Computational Biology*.

Pan, Q.; Shai, O.; Lee, L. J.; Frey, B. J.; and Blencowe, B. J. 2008. Deep Surveying of Alternative Splicing Complexity in the Human Transcriptome by High-Throughput Sequencing. *Nature Genetics*.

Pertea, M.; Lin, X.; and Salzberg, S. L. 2001. GeneSplicer: A New Computational Method for Splice Site Prediction. *Nucleic Acids Research*.

Ruohan, W.; Zishuai, W.; Jianping, W.; and Li, S. 2019. SpliceFinder: ab initio Prediction of Splice Sites Using Convolutional Neural Network. *BMC Bioinformatics*.

Sherstinsky, A. 2020. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Physica D: Nonlinear Phenomena*.

Tahir, M.; Hayat, M.; Khan, R.; and Chong, K. T. 2022. An Effective Deep Learning-Based Architecture for Prediction of N7-Methylguanosine Sites in Health Systems. *MDPI*.

Tayara, H.; Tahir, M.; and Chong, K. T. 2019. iSS-CNN: Identifying Splicing Sites Using Convolution Neural Network. *Chemometrics and Intelligent Laboratory Systems*.

Zuallaert, J.; Godin, F.; Kim, M.; Soete, A.; Saeys, Y.; and De Neve, W. 2018. SpliceRover: Interpretable Convolutional Neural Networks for Improved Splice Site Prediction. *Bioinformatics*.

Author Index