

# Applying New Model Architectures to the Color Context Problem

**Amber Yang**  
Stanford University  
yanga@stanford.edu

**Kasha Akrami**  
Stanford University  
kakrami@stanford.edu

**Vivek Kumar**  
Stanford University  
vivkumar@stanford.edu

## Abstract

As an extension the Colors Homework Assignment, our group was curious to see how LSTMs in the encoder-decoder class, bidirectionality, and fine-tuning of hyperparameters would impact our model’s performance. Our intuition for this derives from the fact that throughout the course, selection of certain model features such as encoder-decoder structuring have direct impact on the model’s performance. Moreover, trade-offs to consider in model design include time complexity, efficiency, and performance. With this in mind, we set out to design a model that optimized for performance over efficiency. We hope that by our model’s updated design, we can present a better-performing approach to the original Colors Homework Assignment.

## 1 Introduction

After completing the color context assignment, there were underlying extensions and applications our group was interested in exploring. Specifically, could a bidirectional LSTM for the encoder-decoder model improve the performance in addition to the other hyperparameter changes in the LSTM model?

As we have learned throughout the course, our choice of model architecture in the encoder-decoder model alongside hyperparameter tuning can have significant impact on model performance. Furthermore, there are other studies that have explored these ideas, including (Reimers and Gurevych, 2017) (who explore fine-tuning of hyperparameters in BERT networks) and (Cho et al., 2014) (who explore the use of LSTM versus GRU in model architecture).

Furthermore, during our original implementation of the problem, we had been keen to explore using bidirectionality within the context of an LSTM encoder-decoder model. Unfortunately, managing the matrix expansion/decompression was too complex within the assignment time-frame, and we decided that it would make for interesting exploration

in a final project. One previous study which had explored this was (Schuster and Paliwal, 1997), which found positive model performance in RNN’s that made use of bidirectionality. This sent a positive signal to our group to explore model performance of LSTM’s with bidirectionality.

## 2 Prior Literature

For background context, the pragmatic color descriptions task is a grounded communication task at its core with the ultimate goal of identifying colors from textual descriptions that draws from the predictions of two recurrent neural network (RNN) classifiers: a speaker and a listener. The problem is first defined in (Monroe et al., 2017), and the ultimate conclusion is that the classifications from the combination of the speaker and listener RNNs are more accurate than the classifiers from which they are built. Furthermore, the proposed model is most helpful in situations where the model must distinguish between similar color hues.

Ultimately, the model presented by Monroe et al. is built around a version of the Rational Speech Acts (RSA) model (Frank and Goodman, 2012) and (Goodman and Frank, 2016). The RSA recursion is defined in terms of the base agents: the pragmatic speaker which produces utterances based on a literal RNN listener, and the pragmatic listener which interprets utterances based on the pragmatic speaker’s behavior. These two agents reason recursively about each other’s expectations and intentions to communicate more effectively than literal semantic agents could. The literal semantic agents are recurrent neural networks (RNNs) that produce and interpret color descriptions in context. These models are learned from data and scale easily to large datasets containing diverse utterances. The most successful model integrates speaker and listener perspectives, and combines predictions made by a system trained to understand color descriptions and one trained to produce them. The linguistic

behavior of the players exhibits many of the intricacies of language in general, including not just the context dependence and cognitive complexity discussed above, but also compositionality, vagueness, and ambiguity.

From the RSA model, it is observed that the largest improvement over the single RNN comes from blending it with an RNN trained to perform the speaker task. Pragmatic reasoning on top of the listener RNN alone also yields improvements. The hard cases in classification comes from : 1) contexts with colors which are very similar, and requires the interpretation of descriptions that convey fine distinctions; and 2) target colors that most referring expressions fail to identify, either due to a lack of adequate description or a consistent bias against the color in the RNN listener.

With background established on the color context problem, there are a substantial range of previous studies on LSTM architecture in encoder-decoder models, usage of bidirectionality in encoder-decoder models, fine-tuning of hyperparameters in model architecture, and more that we knew we could draw from during our research process.

To begin, we explored literature that examined why usage of LSTM’s and GRU’s were better for model performance than standard RNN networks we were familiar with. For this, we turned to (Jozefowicz et al., 2015), where they evaluated thousands of different RNN architectures but ultimately concluded that LSTMs and GRUs outperform almost all RNN models. With this understanding, we then took further dive into the architecture of LSTM’s and their performance. For this, both (Kalchbrenner et al., 2015) and (Greff et al., 2017). These two studies collectively explored ten different LSTM architectures only to discover that the permutations of the vanilla LSTM model did not have a substantial improvement on model performance over vanilla LSTM. The literature review on ideal model architecture for an RNN-type structure for natural language made us feel confident in our task of applying an LSTM encoder-decoder model that would be bidirectional.









When we finally turned to understand how tuning for hyperparameters on an established architecture works, we began with (Greff et al., 2017). Their study discovered in their testing of LSTM architectures that learning rate and network size are the two most crucial hyperparameters for LSTM

models. Furthermore, (Reimers and Gurevych, 2017) provided useful approaches for finding the optimal hyperparameters for deep LSTM networks for natural language classification tasks. We consequently adopted some of these strategies when doing our own hyperparameter tuning, with a random search over a finite search space for a number of classic model hyperparameters.

### 3 Data

The dataset we use follows a standard structure that was overviewed in (Monroe et al., 2016). It is based on the Standard English Colors in Context Corpus (SCC).

The corpus itself is based on a two-player interactive game, where a speaker is privately assigned a target color and asked to produce a description, and a listener aims to identify the speaker’s target. In the data, each row is composed of three colors, with the target color being boxed and the other two colors being distractors. Next, each row of data could follow one of three conditions: far, split, close. In the far case, all three colors are far apart in color space. In the split case, the target color is close to one of the distractors. In the close case, the target color is similar to both distractors. A snippet of the data corpus can be seen in Figure 1.

	Context		Utterance
1.			darker blue
2.			Purple
3.			blue
4.			blue

(Monroe et al., 2017)

Figure 1: Examples of color reference in context, taken from new corpus. The target color is boxed. The speaker’s description is shaped not only by this target, but also by the other context colors and their relationships.

In order to avoid excessively long training run times, we focused on using the two-word-only subset of the SCC, which amounted to 13890 examples in the dev set. Although this is not a generalizable dataset since most examples in the SCC are either longer or shorter descriptions, it should suffice for debugging and development. All results are

ultimately reported from testing on the full SCC dataset.

Now, let's dive into the dataset generation pipeline from the SCC. First, we extract the raw color, which will serve as the inputs, and raw texts from the SCC. However, the raw texts are not in string format, so we utilized a sub-word level tokenizer that splits the string on whitespaces and adds [START] and [END] symbols to the output list of tokens.

Only the training set is used to create the model's vocabulary, and a random train-test split is used for the development runs.

## 4 Model

The heart of the model we are using to solve this problem is a standard encoder-decoder model. Encoder processes the color contexts as a sequence where as Decoder is a neural language model whose initial hidden representation is the final hidden representation of the Encoder.

### 4.1 Baseline Model

For the baseline model, we are using GRU based encoder-decoder as defined in the *colors\_overview.ipynb*. First created as a model improvement of RNNs, GRUs aim to solve the vanishing gradient problem that can occur with larger models (Kyunghyun Cho, 2014), which make it a promising baseline candidate model. Furthermore, since the problem can become complex quickly with longer text descriptions in the SCC dataset, a slight improvement on the RNN model makes sense to use. The baseline encoder and decoder

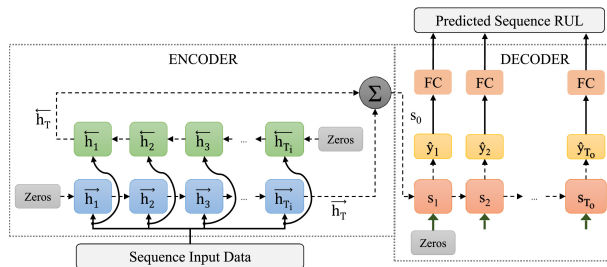


Figure 2: Base Model architecture of Seq2Seq prediction using encoder-decoder

have only one hidden layer. As a first next step, we propose to make it a multi layer model for both the GRU encoder and decoder of the same depth.

### 4.2 GRU Encoder-Decoder Model

The first model we extended to investigate is multi-layer, bi-directional GRU. As our baseline model is one layer unidirectional GRU, this seemed like very first step. In order to find optimal model, we searched through multiple models to find best hyperparameters of hidden\_dim and num\_layers of encoder and decoder. On top of hyperparameters, we used bidirectionality first only on encoder, and then on both encoder and decoder to understand significance of bidirectional nature of GRU based model. The evaluation metrics for search across these models are *bleu* and *listener\_accuracy*.

### 4.3 LSTM Encoder-Decoder Model

The next set of models in our exploration involves using multi layer, unidirectional bidirectional LSTM instead of GRU for both the encoder and the decoder model architecture. Referencing the work of (Jozefowicz et al., 2015), where various GRU and LSTM architectures were explored on the same base problem, it seemed reasonable to explore an LSTM as a next step from the baseline model.

We expect to benefit from bidirectional nature of model to capture information from both sides: left to right and right to left. At the same time, by adding more depth to layers, we want to allow the model to fit to more complex functions.

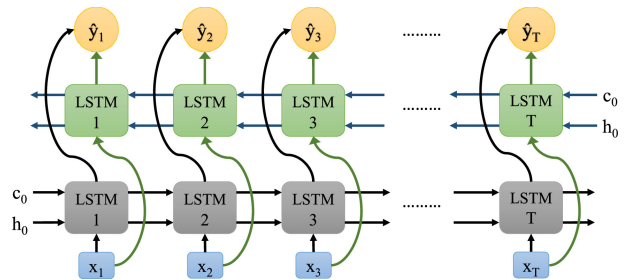


Figure 3: Architecture of bidirectional LSTM

On top of adding bidirectionality to the LSTM, we plan to experiment with multiple parameters of the model like embedding size, hidden size, and number of hidden layers driven by desire to achieve better score and determine optimal model complexity for the SCC dataset.

### 4.4 Transformer-based Model

An additional model that we investigated is a Seq2Seq prediction using encoder decoder models based on transformer architecture models, like BERT.

(Sascha Rothe, 2020) showed the effectiveness of initializing sequence-to-sequence models (like RNNs) with pretrained base models for sequence generation tasks. (Yang Liu, 2019) then proved that this type of architecture can be leveraged with two pretrained BERT models as the encoder and decoder, and that this encoder-decoder model resulted in better results than a classic RNN architecture that is not pretrained.

The focus of our work is to understand if it is possible to adapt these models with fine tuning and whether or not it performs better than sequence to sequence models like LSTM and GRU for the task at hand.

Specifically, our model uses a sequence-to-sequence architecture with encoder-decoder that are made up of transformer layers. For the encoder, a BERT transformer layer is used with a GELU activation function instead of a RELU (Dan Hendrycks, 2019). The decoder is the same except that an encoder-decoder attention mechanism is added as was done in (Sascha Rothe, 2020). The model is made up of 12 layers with each hidden layer consisting of 768; a filter size of 3072, and 12 attention heads.

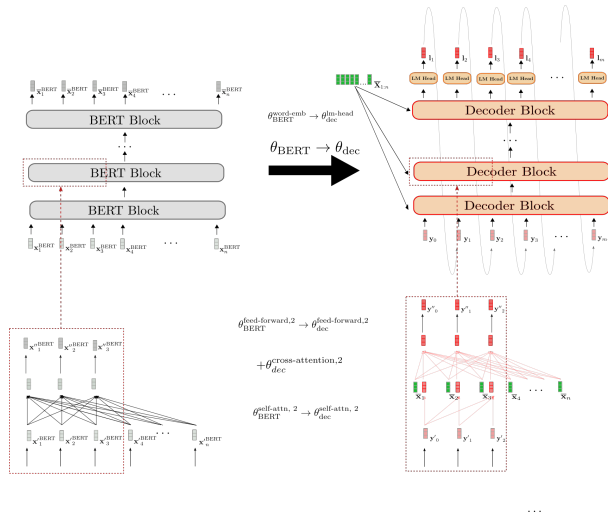


Figure 4: Model architecture of pre-trained transformer-based encoder-decoder

Furthermore, we use BERT checkpoints in our model training. We do this by first tokenizing our data the way that the BERT pre-trained vocabulary is tokenized. The checkpoints have a vocabulary size of approximately 30,000 word pieces, and BERT trains positional embeddings for up to 512 positional sequences (which is the maximum input and output length for all experiments).

Another extension of Seq2Seq based model, we investigated to do is to perform prediction classification based on multimodal BERT where word\_seq for dataset examples is input as text where color\_seq is input as numerical features and together both type of features can be combined to do prediction classification.

## 5 Methods

The experimental approach, including descriptions of metrics, baseline models, etc. Details about hyperparameters, optimization choices, etc., are probably best given in appendices, unless they are central to the arguments.

### 5.1 Data Preprocessing

For all models used in this project, the first step is to tokenize the input by converting it to lowercase, followed by removal of whitespace, punctuations and performing sub-word level segmentation at the end. For eg: an input string 'Darker bluish!' tokenize to {'dark', 'er', 'blu', 'ish', '!'}.

Tokenized input is also added with tokens of start and end symbol.

We then improve the color representation by first transforming the HLS format to the HSV format and apply Fourier transform method of (Monroe et al., 2016) and (Monroe et al., 2017). We create a GloVe embedding based on the model vocabulary on a sub-word level.

Since all models for the color context problem rely on an encoder-decoder (listener-explainer) model, the following high-level workflow of an encoder-decoder model applies to all models used in this project. Using GloVe embeddings, the encoder generates the output and hidden states for the input color\_seqs and word\_seqs. For the decoder, the input vector to the model at each timestep is a concatenation of input token representation with the representation of the target color. The encoder then extracts the target colors from color\_seqs and feeds them to the decoder to predict new sequences based on the color contexts in the input color\_seqs.

### 5.2 Metrics

Evaluation metric of listener accuracy and BLEU score is evaluated on a dev dataset to determine the course of model improvement.

The key metric that we will use to compare the various model architectures is the listener accuracy and BLEU score evaluated on the dev set. Since



this is a classification task, we will create a confusion matrix for the listener neural network for each iteration of our model. This will provide us with a more holistic comparison of how the observed/gold labels compare to the labels predicted by the listener classifier.

Since the transformer-based encoder-decoder model uses the established BERT transformer, the listener accuracy metric that we use for the GRU and LSTM-based models cannot be integrated with the BERT specific outputs. Thus, we will use precision, recall, and F1 score to measure the performance of the BERT-based model. Calculating the precision will give us more information on a per-class basis. It is a straight-forward extension to then take the precision and recall to calculate the F score (another per-class metric).

### 5.3 Baseline Model Parameters

The baseline model consists of using a GRU for the encoder-decoder. We began by using two layers for both the encoder and decoder that are both unidirectional. With this architecture, we tested out hidden layer sizes of 50, 100, 200, and 300 neurons. We then decided to test out encoders and decoders with three, four, and five layers each with 300 neurons for the hidden layer size.

Our next architecture exploration is into the directionality of the encoder and decoder. We experimented with two more model architectures: one with a bidirectional encoder and unidirectional decoder and another with both the encoder and decoder being bidirectional. We then varied the number of hidden layers and the hidden layer size in the same manner as we did for the unidirectional model for both of these multi-directional model architectures.

GloVe embeddings are used for all architectures for the baseline model.

### 5.4 LSTM Encoder-Decoder Model Parameters

As justified in Section 4.2 of this paper, replacing GRUs with LSTMs for the encoder-decoder model is a reasonable next step towards improvement from the baseline model.

First, we used unidirectional LSTMs for the encoder-decoder. We kept the decoder at one hidden layer and varied the encoder with either one or three hidden layers. With the possible combinations that result from changing the number of hidden layers, we varied the size of hidden layers

from 50, 100, 200, and 300 neurons. We made an additional hyperparameter exploration into varying the types of embeddings used. Beyond GloVe embeddings, we tested out the various model architecture combinations with random initial embeddings of multiple sizes: 50, 100, and 300.

Upon finishing this exploration of hyperparameters, we began experimenting with the directionality of the encoder-decoder similarly to what we did in Section 5.3. We tried out the following combinations: bidirectional encoder and unidirectional decoder, unidirectional encoder and bidirectional decoder, and bidirectional encoder and decoder.

We explored hyperparameter adjustments to the three architecture combinations just mentioned in a similar manner to how we explored hyperparameters for the unidirectional LSTM encoder-decoder mentioned earlier in this section with one notable exception. For the LSTM bidirectional encoder and unidirectional decoder, we added explored using 0.1 and 0.2 for the dropout parameter for both the encoder and decoder.

### 5.5 Transformer-based Model Parameters

For a transformer based encoder decoder model, we used BERT2BERT architecture. The max\_length of both encoder and decoder is 512. Changing decoder max\_length to 128 helped in faster training time but no negative effect on the final score. The initial setup code for this work is inspired from Encoder Decoder Models description on huggingface documentation. Pre-processing setup of SCC corpus to provide as input to BERT2BERT model for fine tuning is one of the most challenging task we encountered in this project.

## 6 Results

Below are the results from variations of GRU based encoder decoder models. As discussed in Section 5.3, we modified hidden\_dimension, num\_layers and directionality of each of the encoder and decoders.

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
50	2	50	2	0.6609	0.7984	Glove
100	2	100	2	0.6661	0.8172	Glove
200	2	200	2	0.6675	0.8226	Glove
300	2	300	2	0.6602	0.8246	Glove
300	3	300	3	0.6620	0.8232	Glove
300	4	300	4	0.6539	0.7904	Glove
300	5	300	5	0.5341	0.3346	Glove

Figure 5: Results for GRU with Encoder-unidirectional, Decoder-unidirectional

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
300	4	300	8	0.4266	0.3562	Glove
300	2	300	4	0.6637	0.8103	Glove
200	2	200	4	0.6502	0.7984	Glove
100	2	100	4	0.6615	0.8114	Glove
50	2	50	4	0.6556	0.7918	Glove

Figure 6: Results for GRU with Encoder-bidirectional, Decoder-unidirectional

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
300	4	300	4	0.3892	0.7322	Glove
300	2	300	2	0.3892	0.7391	Glove
200	2	200	2	0.3892	0.7671	Glove
100	2	100	2	0.3892	0.7535	Glove
50	2	50	2	0.3892	0.7296	Glove
200	3	200	3	0.3895	0.7377	Glove
300	3	300	3	0.3632	0.7475	Glove

Figure 7: Results for GRU with Encoder-bidirectional, Decoder-bidirectional

Below are the results from variations of LSTM based encoder decoder models. As discussed in Section 5.4, we modified hidden\_dimension, num\_layers and directionality of each of the encoder and decoders. We also explored the impact of Glove embedding vs. random initial embedding.

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
50	1	50	1	0.6619	0.8010	Glove
100	1	100	1	0.6640	0.8100	Glove
200	1	200	1	0.6638	0.8172	Glove
300	1	300	1	0.6651	0.8094	Glove
50	3	50	1	0.6598	0.7907	Glove
100	3	100	1	0.6624	0.7987	Glove
200	3	200	1	0.6603	0.8085	Glove
300	3	300	1	0.6626	0.8126	Glove
300	3	300	1	0.6647	0.8094	random-50
300	3	300	1	0.6603	0.8091	random-100
300	3	300	1	0.6584	0.8056	random-300
200	3	200	1	0.6660	0.8137	random-50
200	3	200	1	0.6592	0.8154	random-100
200	3	200	1	0.6625	0.7950	random-300
100	3	100	1	0.6635	0.8151	random-50
100	3	100	1	0.6608	0.8048	random-100
50	3	50	1	0.6572	0.7766	random-50
50	3	50	1	0.6633	0.7947	random-100

Figure 8: Results for LSTM with Encoder-unidirectional, decoder-num\_layers being one, Decoder-unidirectional

Figure 12 shows the the result of Sequence based Encoder decoder model based on BERT and details of model is discussed in section 5.5.

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
300	3	300	1	0.6662	0.8200	random-50
200	3	200	1	0.6640	0.8174	random-50
100	3	100	1	0.6635	0.8215	random-50
100	3	100	1	0.6617	0.8103	random-100
50	3	50	1	0.6535	0.7855	random-50
50	3	50	1	0.6607	0.7973	random-100

Figure 9: Results for LSTM with Encoder-bidirectional, decoder-num\_layers being one, Decoder-unidirectional on random initial embedding

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
50	3	50	1	0.6612	0.7933	Glove
100	3	100	1	0.6644	0.8140	Glove
200	3	200	1	0.6626	0.8146	Glove
300	3	300	1	0.6651	0.8226	Glove
50	3	50	6	0.5909	0.5514	Glove
100	3	100	6	0.4266	0.3570	Glove
200	3	200	6	0.6358	0.7751	Glove
300	3	300	6	0.6591	0.8010	Glove
300	3	300	6	0.6367	0.7924	Glove
300	3	300	6	0.6345	0.7835	Glove
300	2	300	4	0.6501	0.8088	Glove

Figure 10: Results for LSTM with Encoder-bidirectional, Encoder-num\_layers being three, Decoder-unidirectional

Encoder - hidden_dim	Encoder - num_layers	Decoder - hidden_dim	Decoder - num_layers	BLEU	Listener - accuracy	Embedding-dim
50	2	50	2	0.4054	0.6599	Glove
100	2	100	2	0.3892	0.6628	Glove
200	2	200	2	0.3892	0.6481	Glove
300	2	300	2	0.3892	0.6689	Glove
50	3	50	3	0.3892	0.5603	Glove
50	4	50	4	0.3892	0.5220	Glove
50	5	50	5	0.3892	0.3706	Glove
50	6	50	6	0.3892	0.3881	Glove
100	3	100	3	0.3892	0.6219	Glove
200	3	200	3	0.3892	0.5603	Glove
300	3	300	3	0.3892	0.6093	Glove
300	4	300	4	0.3892	0.5428	Glove
300	5	300	5	0.3892	0.6119	Glove
300	6	300	6	0.3892	0.6473	Glove

Figure 11: Results for LSTM with Encoder-bidirectional, Decoder-bidirectional

Metric	Score
F1	0.5718
Precision	0.4256
Recall	0.8709

Figure 12: Results for BERT2BERT Encoder Decoder Model, Encoder-max\_length=512, Decoder-max\_length=128

## 7 Analysis

Regarding the GRU model structure, we found that the best performing model was unidirectional in the encoder, unidirectional in the decoder, had 300 hidden dimensions in the encoder, has two layers in the encoder, had 300 hidden dimensions in the de-

coder, has two layers in the decoder, and a GLOVE embedding. Its BLEU accuracy is 66.02% and its listening accuracy is 82.46%.

Shifting over to the LSTM model structure, we found that the best performing model was bidirectional in the encoder, unidirectional in the decoder, had 300 hidden dimensions in the encoder, had three layers in the encoder, had 300 hidden dimensions in the decoder, had one layer in the decoder, and a GLOVE embedding. It's BLEU accuracy was 66.51% and listener accuracy was 82.26%. When looking at the general results of LSTM's performance, we observed a few key trends. First, incorporating more states was better than less in the hidden dimensions of both the encoder and the decoder. Having less states in the hidden dimensions of both the encoder and the decoder corresponded with a lower BLEU and listening accuracy. In addition, having too many layers for both the encoder and decoder did not mean improved BLEU and listening accuracy.

It was interesting to observe that unidirectional performed better than bidirectional in GRU, and like LSTM, incorporating more hidden dimensions in the encoder and decoder generally resulted in better performance.

With fine tuning of BERT2BERT model for encode decoder task on SCC corpus, we observed that integration of new dataset to hugging face infrastructure is not straightforward as it requires custom pre processing of data. It took around 4 hours to train the model for 1 epoch on a Tesla K80 GPU machine, primarily because we had to run with small batch size of 2 to overcome issue of CUDA's insufficient memory errors. We did achieve F1 score of 0.57 with max\_length 512 for encoder and max\_length 128 for the decoder. Training and validation loss are observed to be 0.019 and 0.14 respectively, suggesting model performance overfit to the size of dataset.

## 8 Conclusion

The goal of this project is to explore alternative encoder-decoder model architectures to apply to the color-context model. Specifically, we applied GRUs, LSTMs, and BERT transformer to the encoder-decoder. For the GRU and LSTM-based models, we performed a hyperparameter search exploring bidirectionality, the usage of dropout, embeddings, the ideal number of hidden layers, and the hidden layer size of the encoders and decoders.

We implemented the BERT encoder-decoder model from Hugging Face for this specific problem.

For the GRU and LSTM-based models, our hypothesis that model architectures with more hidden layers and larger hidden layer sizes perform better held true. However, one of our hypotheses regarding bidirectionality of the GRUs and LSTMs did not prove to be true. Postulating that bidirectionality improves the the interconnectivity of the models, we expected the listener accuracy of bidirectional models to be higher. However, we found that the best GRU model architecture was unidirectional in both the encoder and the decoder. The best LSTM model architecture was bidirectional in the encoder and unidirectional in the decoder. These results should be caveated with the reality that we have a small sample of training that we ran to prove that this is conclusive evidence, but it is an interesting finding.

Since the established outputs of BERT are difficult to align with the listener accuracy we measured for the GRUs and LSTMs, it was difficult for us to do a side-by-side comparison. However, we found that the BERT model's recall score is decently high, with a low precision score, and an average F1 score. The low precision score can be attributed to the fact that we used two distractors and only one target calss. Thus, the sizes of our classes are imbalanced, and we observe a low overall recall. We are sharing notebooks for all the code that we developed for evaluating above models. We hope that work of SCC corpus on Transformer based models like BERT can be extended further to align model output to evaluation metric of RNN based encoder decoder model in this case.

## Known Project Limitations

A known limitation of this project (and one that we were clear about since our literature review) is that our models take in data of a very specific structure and may not handle other data sets that are not adapted to this structure. Specifically, like the data we worked with in the original homework assignment, the model takes in three colors per row, with the target color being boxed. Had their been more than three colors or more than one target color, our model would not have performed successfully. The alleviation for this would be to build a more adaptable model architecture, which is something future extensions of this project could do.

## Authorship Statement

Collectively, all authors of the paper worked on the baseline model and the subsequent hyperparameter exploration of the baseline GRU encoder-decoder model.

Amber and Kasha worked on the LSTM bidirectional encoder and unidirectional decoder. Vivek worked on the bidirectional encoder and bidirectional decoder of GRU and LSTM.

Vivek implemented the BERT transformer encoder-decoder model.

Reporting of the results, analyses, and trends of the results as they relate to the models were done by all authors.

## References

- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. [Learning phrase representations using rnn encoder-decoder for statistical machine translation](#).
- Kevin Gimpel Dan Hendrycks. 2019. Bridging non-linearities and stochastic regularizers with gaussian error linear units.
- Michael C. Frank and Noah D. Goodman. 2012. [Predicting pragmatic reasoning in language games](#). *Science*, 336(6084):998–998.
- Noah D. Goodman and Michael C. Frank. 2016. [Pragmatic language interpretation as probabilistic inference](#). *Trends in Cognitive Sciences*, 20(11):818–829.
- Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. [LSTM: A search space odyssey](#). *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures.
- Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. [Grid long short-term memory](#).
- Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho, Dzmitry Bahdanau. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.
- Will Monroe, Noah D. Goodman, and Christopher Potts. 2016. [Learning to generate compositional color descriptions](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2243–2248, Austin, Texas. Association for Computational Linguistics.
- Will Monroe, Robert Hawkins, Noah Goodman, and Christopher Potts. 2017. [Colors in context: A pragmatic neural model for grounded language understanding](#). *Transactions of the Association for Computational Linguistics*, 5(0):325–338.
- Nils Reimers and Iryna Gurevych. 2017. [Optimal hyperparameters for deep lstm-networks for sequence labeling tasks](#).
- Aliaksei Severyn Sascha Rothe, Shashi Narayan. 2020. Leveraging pre-trained checkpoints for sequence generation tasks.
- M. Schuster and K.K. Paliwal. 1997. [Bidirectional recurrent neural networks](#). *IEEE Transactions on Signal Processing*, 45(11):2673–2681.
- Mirella Lapata Yang Liu. 2019. Text summarization with pretrained encoders.

## A Example Appendix

This is an appendix.