Insemble: A Shared-Parameter Ensembling Technique for Question-Answering

Stanford CS224N Default Project

Luke Hansen
Department of Computer Science
Stanford University
lrhansen@stanford.edu

Justin Lim
Department of Computer Science
Stanford University
jlim23@stanford.edu

Abstract

In this paper, we explore a variety of techniques that perform better than a baseline BiDAF model's question-answering performance on SQuAD 2.0, as measured by EM and F1 scores. We first improve upon the baseline BiDAF scores and training speed by replacing the Bi-directional LSTM with a Gated Recurrent Unit (GRU) and adding a character-level embedding layer. We found this model to better than a BiDAF with self-attention and a QANet. We propose a technique called Insemble: a variation of ensembling that consists of multiple submodels with some shared parameters. We used three GRU with Character Embeddings models with sligthly different hyperparamters and architectures as our submodels. Insemble performed better than a standalone GRU with Character Embeddings (even with the same number of parameters) and a vanilla ensembling implementation (Three GRU with Character Embeddings that do not share parameters). It achieved test set F1 and EM scores of 65.46 and 62.40, respectively. We also found that, for Insemble, higher scores only scale with an increased parameter size to a certain point.

1 Key Information to include

- TA mentor: Vincent Li
- External collaborators: None
- Sharing project: None

2 Introduction

As human interaction becomes increasingly intertwined with machines, one of the largest areas of research is Machine Comprehension and Question Answering. With the conception of state-of-the-art NLP architectures like Transformers, models have had impressive results on a variety of text-based tasks [1].

Increasing the size of language models reliably increases their performance, given there is enough data [2]. For some tasks, however, there is not enough data to take advantage of the available computing power, as models which are too large can be prone to overfitting. Ensembling (training multiple networks and combining the output) has helped avoid this problem because more compute power can be used without the risk of overfitting [3]. However, this technique is also resource expensive since multiple models need to be trained.

Intra-ensemble is a technique that capitalizes on the benefits of ensemble in only one network through having various submodels that share many parameters, making Intra-ensemble more space efficient than a naive ensemble [4].

Given the success and efficiency of Intra-ensemble on an image recognition task, we wondered if we could use similar principles as those used in Intra-ensemble to improve the performance of models on a QA task. We created Insemble: a language processing architecture inspired by the parameter sharing technique proposed by the Intra-ensemble paper (see Figure 1). We explored various submodel candidates for Insemble, including a provided baseline BiDAF [5]; a BiDAF with character embeddings and a GRU RNN [6] instead of an LSTM [7]; a BiDAF using self-attention for one of its encoders instead of an RNN; and a QANet[8]. We tested these model's question answering performance on SQuAD 2.0 [9] and measures their EM and F1 scores.

We found the BiDAF with a GRU RNN and character embeddings performed the best out of our implemented models, so we used it as the backbone submodel in Insemble (the far left submodel in Figure 1). Our final model consisted of three submodels: a GRU-character-embeddings model, a LSTM-GRU-encoder model, and a GRU-LSTM-encoder model. The output of the final model was the average of the output of these three submodels. These three models also shared some of the encoder layers, allowing for a combination of information and a reduction in the total number of parameters, similar to the approach of Intra-ensemble.

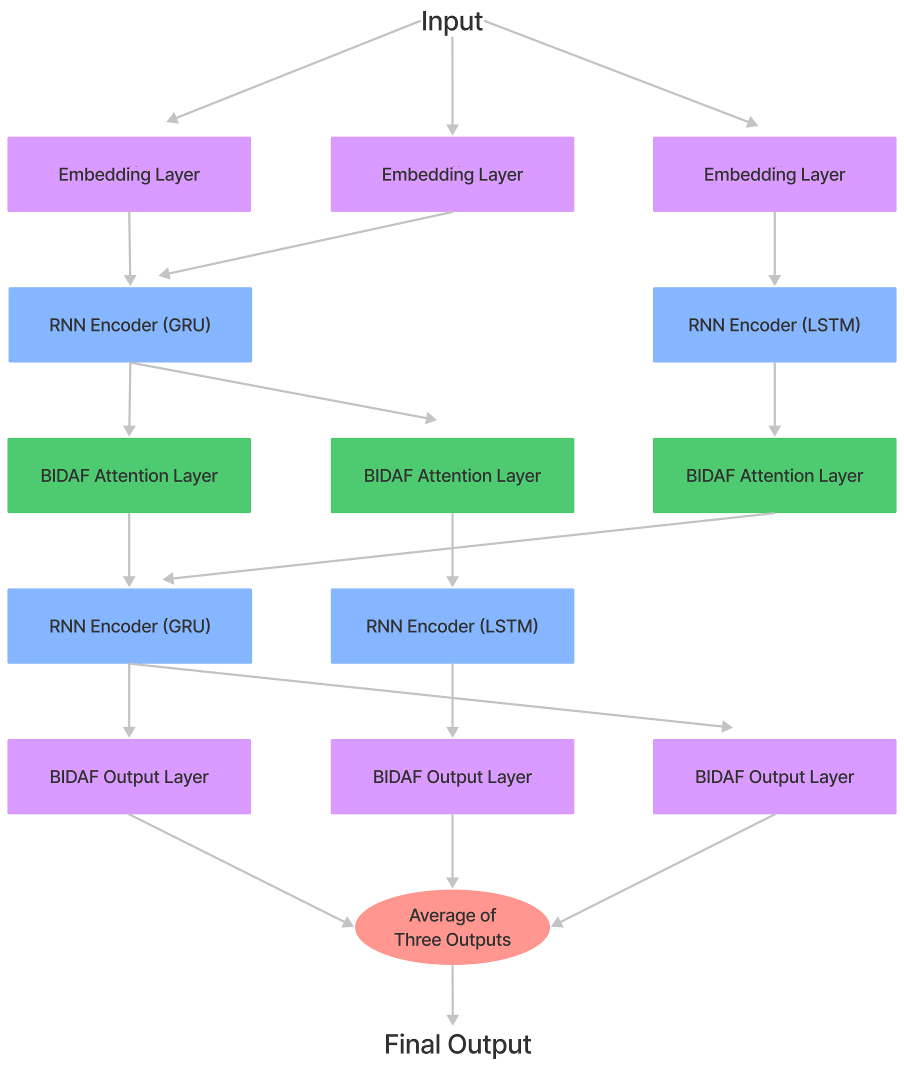

Figure 1: Insemble Model: A Shared-Parameter Ensemble Network

This model performed better than a naive ensemble strategy that does not share parameters, and it performed better than using one of the submodels as a standalone model (with the same number of parameters as in the Insemble). This finding suggests that training end-to-end submodels that share parameters is an effective way to decrease the total number of parameters in ensemble models without hurting performance.

3 Related Work

3.1 Intra-ensemble:

Ensembling is a standard technique used in many fields of machine learning, based on the idea that model performance can be enhanced by combining several base models and pooling together their outputs. However, ensembling requires the training of several deep neural networks, a resource intensive operation. Intra-ensembling is an end-to-end ensemble strategy that uses stochastic channel recombination operations to train sub-networks concurrently as one network while increasing the diversity of the sub-networks. A benefit is that there is only a small increase in parameter size since the sub-networks share a majority of their parameters. Intra-ensemble was originally designed for an image task, and experiments were done to show that it performed well on simple and complex tasks using datasets of various image sizes.

3.2 Self-attention:

Self-attention is a module that allows the inputs to a model to interact with each other and decide which parts of the input should recieve more attention. Self-attention is a major part of Transformers, an extremely successful model architecture that discards recurrent and convolution layers entirely.

3.3 QANet:

QANet uses convolutions and self-attention as encoder blocks with no recurrent nature (unlike BiDAF). It also encodes positional embeddings to make up for its lack of recurrence. QANet has been shown to be very powerful in reading comprehensions tasks and was the state-of-the-art before the advent of large transformer models.

4 Approach

Insemble uses a traditional QA architecture as a submodel. To choose a submodel, we experimented with the following candidates:

4.1 Vanilla BiDAF:

The baseline BiDAF model is provided by the Stanford CS224n staff. The embedding, encoder, attention, modeling, and output layers are implemented as described in the original paper [5] with the exception of character embeddings, which were excluded in this implementation.

4.2 GRU and character embeddings:

On top of the vanilla BiDAF model, we re-implemented a character-level embedding layer with batch normalization. This added layer allows the model to understand words based on their internal representations. It greatly improves the model's ability to handle unencountered words, whether they are misspelled or newly invented. The character embedding vectors and indices are already included in the starter code, so we load them in and pass them through a convolutional layer. After several dimension permutations and running through a dropout layer, we concatenate the output to the original word embeddings, and feed it into the highway network. The result now continues through the subsequent original procedures. We expected this strategy to improve our overall F1, EM, and AvNA scores.

We also replace the LSTM encoders of the original BiDAF with GRU encoders, which has fewer parameters due to the lack of an output gate.

4.3 Self attention encoder (with char embeddings):

We replaced the first encoder layer of the BiDAF model with a layer that utilized self-attention. This new encoder layer performed self-attention on the input, and then passed the result to a feedforward network. The baseline version already includes a bi-directional attention flow that allows attention flow from the context to the query (C2Q) and from the query to the context (Q2C). The replaced block adds a layer of context-to-context attention and query-to-query attention.

4.4 QANet implementation (with char embeddings):

As described in the Related Works section, the QANet model does not use recurrent networks unlike BiDAF. Instead, it uses convolutions and multi-headed self-attention in the encoder blocks. We implemented this model in the same way as described in the original QANet paper [8]. We did not implement this with leaky RELU, a tactic that prevents dying RELU [10].

Once the top performing submodel has been identified, variations of it were used for the Insemble network. Ensemble networks tend to work best when each submodel has a good representation of the data, but in a different way. If all the models are the same, there is no benefit to using more of them. We expected a similar principle to apply for for our network. We did not use stochastic channel recombination like the original Intra-ensemble paper (which was used to increase submodel diversity), so we increased submodel diversity by using different RNNs and hyperparameters for each submodel.

One minor optimization that we added was the use of Adam in place of the AdaDelta optimizer [11]. Adam uses SGD with momentum and adaptive learning rates and has been empirically proven to more efficiently converge faster. Thus, we expected it to slightly improve training times and push our model out of local minimums, helping us achieve higher scores.

5 Experiments

5.1 Data

The Stanford Question Answering Dataset [12] (SQuAD 2.0) will be used to train and test the models. The dataset consists of 150k questions about context paragraphs drawn from Wikipedia and comes in the form of context, question, answer triplets. The question, or query, is the question to be answered based on the context. The answer is an excerpt of text of the context. All answerable questions in the training set has one answer per question, but the dev and test sets have three answers per question, provided by crowd workers. SQuAD 2.0 expands on the original SQuAD dataset by adding unanswerable questions, in which case the model should abstain from answering when it determines that the context does not support an answer.

5.2 Evaluation method

We use two metrics to evaluate our model and compare its performance to other models. The first is the Exact Match (EM) score, which is the percent of instances the model output exactly matches the ground truth answer. The second is the F1 score, which reflects the quality of the model as a function of its precision and recall. The precision is measured by whether or not model output is a subset of the tokens of the ground truth answers. The recall is what percent of the ground truth tokens that our model outputted. To help monitor our model training, we also keep track of Answer vs No Answer (AvNA), which measures our model's classification accuracy when considering its answer (any span of text) versus no answer predictions.

5.3 Experimental details 1

All candidate models were trained with a hidden size of 100 until NLL, F1, EM, and AvNA scores plateaued. All models had a learning rate of 0.5 and a drop probability of 0.2, except the QANet which had a learning rate of 0.1 and drop probability of 0.1.

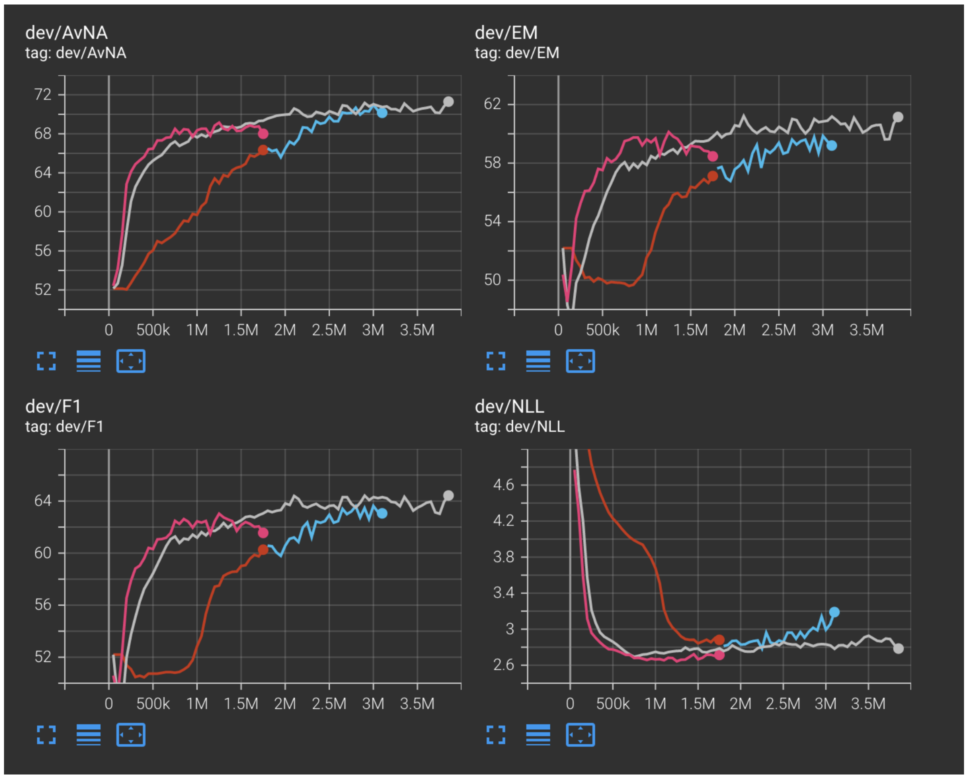

Figure 2: AvNA, EM, F1, and NLL training trajectories of GRU and character embeddings (grey), self attention encoder (pink), and QANet (red and blue–The line is two colors because it was trained in two training periods).

Model	**NLL**	**F1**	**EM**	**AvNA**
BiDAF	3.31	60.18	56.53	67.23
GRU	3.11	60.76	57.22	67.47
GRU + Char-Embed	2.70	64.32	61.1	71.1
Self-Attention Encoder	2.65	62.97	60.01	69.06
QANet	2.75	63.63	60.85	70.98

Table 1: Comparison of submodel performance on dev set

5.4 Results 1

Our models' performance on the dev set can be seen in Table 1, and the training trajectories of the candidate models can be seen in Figure 2. The model using character embeddings and GRU RNN performed the best. This observation was unsurprising because of past work indicating that GRUs are better in this context. This model also was slightly faster to train. However, we were surprised by the QANet's unexpectedly low performance. Given the limited credit count and the extensive time needed to train the QANet (around 14 hours), we decided to focus on testing the Insemble model instead of re-implementing the QANet. This result will be reexamined in the Analysis section.

5.5 Experimental details 2

The model which used a GRU encoder and character embeddings achieved the best dev results, so we decided to use it as the submodel for Insemble. We had three submodels, as depicted in Figure 1. The second and third submodels (from the left in the picture) used LSTMs for their encoder blocks to increase the diversity of representations for each of these submodels. The models were trained with different dropout rates, (0.2; 0.1; 0.2) respectively. Insemble's training process can be seen in Figure 3.

We also trained two models to serve as controls: 1) GRU (225), a standalone model with the same architecture as one submodel (GRU) and a similar number of parameters to the smallest Insemble, allowing us to control for the number of parameters and 2) a vanilla ensemble network (three GRU models with the same hyperparameters) to control for having multiple networks. These controls allowed us to isolate the effects of sharing layer parameters, and having slight variations within the submodels. All these models were trained with a learning rate of 0.5, and a drop probability of 0.2 (unless otherwise specified).

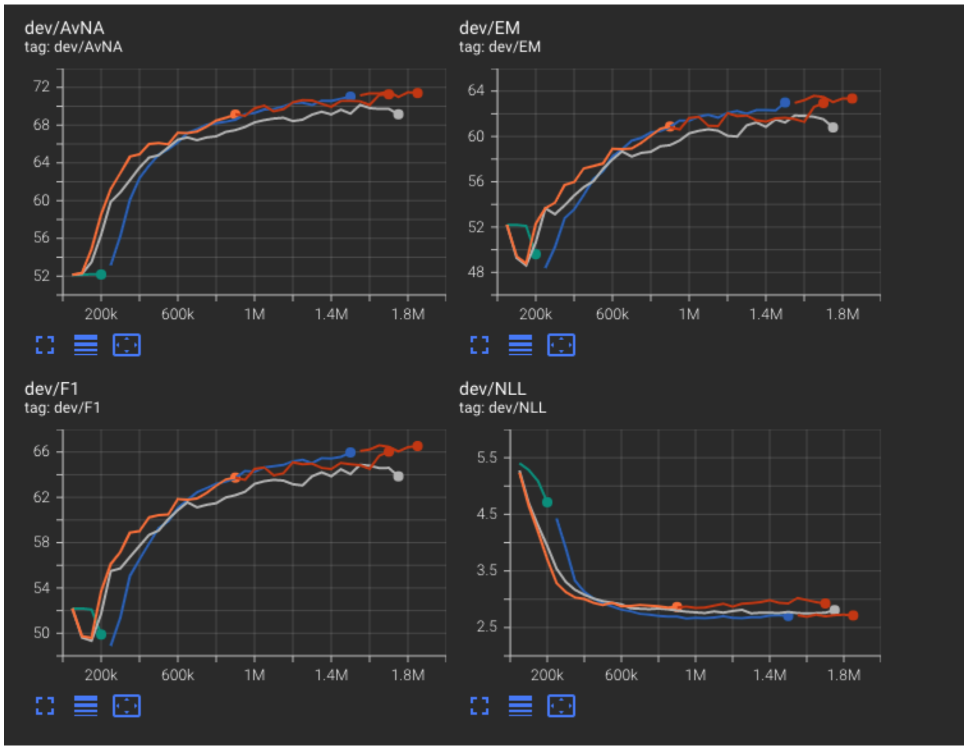

Figure 3: AvNA, EM, F1, and NLL training trajectories of varioius sizes of Insemble: 75 hidden size (grey), 100 hidden size (blue+red), and 150 hidden size (orange+red).

5.6 Results 2

The scores of various Insemble sizes and the controls can be seen in Table 2. The smallest Insemble model allowed for better representations of the data than using a single model with a comparable number of parameters. This single model quickly started to overfit the data. Due to the overfitting, we opted not to test comparable models for the Insemble models with more parameters. Furthermore, Insemble also outperformed the vanilla ensemble model.

Increasing the number of model parameters increased the performance of the model, to an extent–increasing the hidden size from 75 to 100 increased the F1 and EM scores by 1.72 and 1.75, respectively. However increasing the hidden size from 100 to 150 hurt the models performance, lowering the F1 and EM scores by 0.56 and 0.64, respectively.

The Insemble model with a hidden size of 100 achieved the best dev set score, so we ran it on the test set. It achieved F1 and EM scores of 65.46 and 62.40, respectively.

Model	NLL	F1	EM	AvNA
GRU (225) (control)	2.92	61.34	57.67	70.14
Vanilla Ensemble (100) (control)	2.80	64.51	61.44	69.80
Insemble (75)	2.76	64.87	61.84	70.14
Insemble (100)	2.73	66.59	63.59	71.35
Insemble (150)	2.93	66.03	62.95	71.27

Table 2: Comparison of Insemble models with different number of parameters (number in the parentheses represents hidden size value).

6 Analysis

Using a GRU RNN in place of an LSTM immediately improved performance and decreased training times. The addition of character-level embeddings further improved the model's handling of unencountered words and improved the AvNA score. These results were expected, however, our QANet had an unexpectedly low performance. We suspect there might have been a problem with the implementation, specifically in that we used a normal RELU rather than a leaky-RELU, which has been shown to effectively prevent dying gradients. The GRU RNN with character embeddings also outperformed our BiDAF model with self-attention with an F1 score difference of 1.35.

Even though the BiDAF model with self-attention had a lower NLL than the GRU with character embeddings model, it performed worse in terms of F1, EM, and AvNA. We speculate that the self-attention caused the model to predict no-answer more often, causing the faster drop in NLL and faster increase in AvNA, as seen in figure 2. However, this optimization of AvNA and NLL did not help to improve the F1 and EM scores, giving the self-attention model lower scores than the GRU with character embeddings model.

We found that our smallest Insemble (hidden size of 75) outperformed our control models, demonstrating that it improved performance without increasing the number of parameters. However, Insemble's effectiveness did not indefinitely scale with its number of parameters, as the model's score decreased once we got to a hidden size of 150. We believe this is a result of the model's increased representational capacity being underutilized because there was an insufficient amount of data.

Furthermore, we split the Insemble results by question type (How, What, Why, Which, Who, Where, When, Other) as seen in figure 4. All three models performed the best on "when" questions and the worst on "how", "why", and "other" questions. This observation makes sense intuitively, because "when" questions often have more straightforward answers than "how" and "why" questions. All versions of our Insemble perform poorly on "other" questions, which seems to be a direct result of the low number of training examples in this category. Figure 4 also shows that Inesemble 100 has a very low EM but a high F1 score for "why" questions, supporting the idea that it is more difficult to get the perfect answer on "why" questions.

7 Conclusion

In this project, we found that the following strategies performed better than the baseline BiDAF model: replacing the LSTM RNN with GRU RNN; including character embeddings; using self-attention instead of the LSTM RNN; and using QANet model instead of the BiDAF model. Overall, Insemble had promising results–not only did it transfer well from the image task to a reading comprehension task, it performed better than any of the models individually, even considering models with approximately equal parameter sizes. It also outperformed a vanilla ensemble network (without parameter sharing), despite having fewer parameters. Diversifying the submodels by using slightly different encoder blocks was a successful method, but one avenue of further work would entail implementing the a similar operation to the stochastic channel recombination used in the original Intra-ensemble paper. Another avenue of future work would be to successfully train a QANet model, which is known to be more powerful than a BiDAF model, and use that as the submodel for our Insemble. Our experiments also showed that Insemble's performance only scaled with parameter size to a certain extent. A hidden size of 100 was our best performing model, but increasing the size to

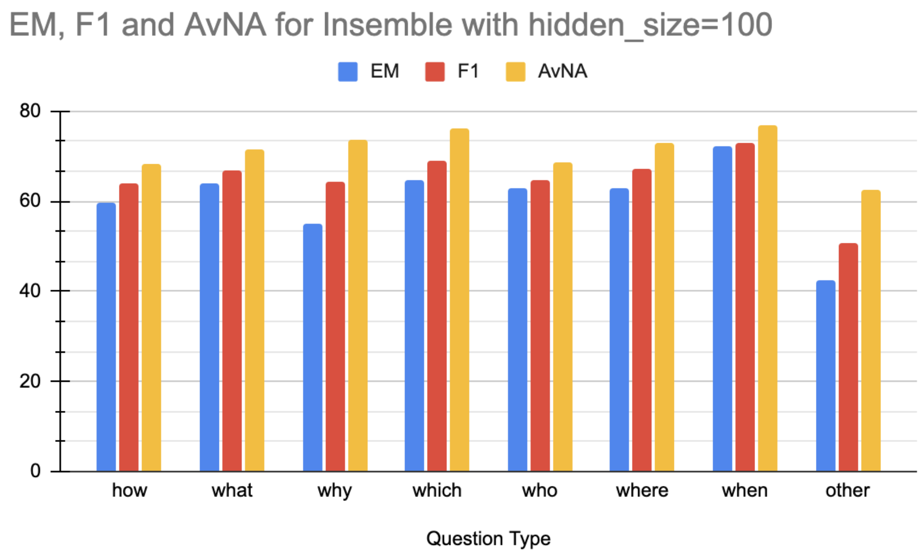

Figure 4: EM, F1, AvNA scores for Insemble-100 separated by question type

150 hurt its performance. Ultimately, we discover that Insemble is a useful technique for reducing the parameter size (compared to other ensemble models) while maintaining high scores.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[2] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021.

[3] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[4] Yuan Gao, Zixiang Cai, Yimin Chen, Wenke Chen, Kan Yang, Chen Sun, and Cong Yao. Intra-ensemble in neural networks. *CoRR*, abs/1904.04466, 2019.

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

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

[8] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.