Predict2Tensor

A Fast, Tensorflow-Based Neural Network for predicting protein secondary structure

Theodore Vlahogiannis

May 7, 2021

1 Abstract

Predict2Tensor is a modern neural-network method for predicting 8-class protein secondary structure. It has an average accuracy of 86%, and 88% percent of predictions made score above 70% accuracy. Additionally, due to the fact that it does not use homology methods like established methods such as JPred and PSIPred, it is incredibly fast, being able to calculate queries in as low as two hundred and fifty milliseconds and the largest queries taking less than 10 seconds. It is also highly scalable. Predict2Tensor can be accessed to submit queries via a frontend application at predict2tensor.com. The entire codebase is freely accessible under the MIT license.

2 Introduction

Though three dimensional structure prediction is currently a hotspot in bioinformatics with projects such as AlphaFold making massive strides, the prediction of secondary structure has fallen out of interest to many, likely due to the fact that it lacks a direct connection to concepts such as drug targeting and mechanisms of action. Secondary structure can be just as important though, as it not only assists in tertiary structure prediction methods such as *de novo* folding and homology modeling, but are also crucial parts of domain identification, and structural superposition calculation. Nonetheless, the field has gradually been increasing in accuracy since the late 1980s, when predictions were only around 50% accurate. Now, predictions can be made with an average accuracy in the high 80%s, with the best techniques getting results in the 90s. Despite this, the playing field is rich with homology based methods or dated neural network frameworks which are slow, unable to be hardware accelerated, and not as extensible as modern technologies. This is especially relevant considering the current explosion of artificial intelligence and neural network technology and the dedicated silicon that is being released at breakneck pace, making ambitious, complex approaches all the more realistic. Predict2Tensor is an approach to filling this niche by developing a modern, bidirectional-LSTM-based approach to predicting secondary structural elements using the industry-standard, flexible, portable Tensorflow framework. Tensorflow is a machine learning library that automates the construction and tuning of dense graphs of algebraic objects known as tensors. With the programmer's direction, Tensorflow can automatically tune the parameters assigned to the tensors in the graphs, in order to optimize them in a way that makes their predictions closer and closer to actual values. Tensorflow has been ported to work with a diversity of hardware and programming languages, and receives frequent feature updates, making it perfect for a field with room to grow such as secondary structure prediction, where the estimated limit of 88%. is yet to be hit. [13]

As it stands, there are two notable, established secondary structure prediction servers: PSIPred, and JPred. Both utilize neural network technologies, though dated. PSIPred uses a Recurrent Neural Network, a method where

prediction later in a sequence is "backpropagated" to earlier in the prediction. However, due to the computational complexity and age of PSIPred, this approach works poorly on longer sequences due to the vanishing gradient problem, where these backpropogations essentially round to 0 as they move father away. As a result, sequences must be broken into multiple fragments, resulting in the loss of information outside each fragment. The fragment size for PSIPred was formerly 17 amino acids, but was recently increased to 33. PSIPred has an average accuracy of 84.2%.

On the other hand, JPred uses a combination of two Convolutional Neural Networks. Convolutional Neural networks utilize a sliding window known as a convolution kernel . The drawback of such methods is that they lose positional information, and are unable to recognize minute detail due to the fact that they aggregate the window into a single value. In the case of JPred, two convolution kernels are used, one of size 17 and one of size 19.[2] JPred is observed to have an average accuracy of 82.0%, with proteins 5% solvent-accessible having an average accuracy of 90%. [3] Both methods also utilize some form of homology information in their predictions.

Though it would appear that the world of secondary structure prediction is already highly competitive, there are some justifications for the introduction of a new method. The first issue is that PSIPred and JPred only make 3 class predictions: α helix, β sheet, and random coil. Though helpful, these limited predictions are not always as useful when considering the pace of modern bioinformatics tasks such as alignment and tertiary structure determination, which can help from a more diverse classification. The other reasoning is due to the fact that neural networks have had a resurgence since the methods used by these two established methods, most notably in the introduction of LSTM networks. LSTM, or Long Short-Term Memory, networks have been designed to challenge the aforementioned vanishing gradient, and thus introduce the ability to consider sequences the length of a typical polypeptide in predictions.[14] Thus, Predict2Tensor aims to tackle these two avenues by offering a bidirectional-LSTM based network which can predict the 8 DSSP classes of secondary structure, and, instead of being developed from the ground up like JPred and PSIPred, can take advantage of existing, modern neural network technologies such as Tensorflow to utilize their optimization, support, and the vast resources available for them, as well as enabling the network to rapidly integrate emerging technologies.

3 Methods

3.1 Data Source

To train the neural network, a local copy of the PDB was made based on the Jan 05, 2021 snapshot, which can be downloaded using the offical rsync script.[11] Using the PDB's REST API, two lists were compiled: one with the PDB ID of all structure entries which contain only amino-acid entities, and the subset of those which contained only a single amino-acid entity. From the first list,

each PDB file was retrieved off-disk, the primary sequence was extracted, and the secondary structure derived. In order to derive the secondary structure, the DSSP algorithm was employed. DSSP is a standard algorithm for determining secondary structures using atomic coordinate information. DSSP makes 8-class predictions according to the following table. [7, 6]

Code	Element
H	α-helix
B	residue in isolated b-bridge
E	extended strand, participates in β ladder
G	3-helix (3_{10} helix)
I	5 helix (π-helix)
T	hydrogen bonded turn
S	bend
-	unstructured loop

Figure 1: The codes implemented in the DSSP algorithm. "Unstructured coil" terminology is used at the behest of the DSSP developers, but is also known as "random coil".[7, 6]

These two data elements, the primary and secondary sequence, were associated with the PDB file and chain they were built from, and then stored. To avoid overfitting, the dataset was cleared of redundancy, using the aforementioned second list to automatically accept monomeric proteins, and then otherwise iteratively determine unique chains. By doing this, the number of chains was reduced from around four hundred thousand to around two hundred and two thousand.

The next data parsing step undertaken was to make the dataset representable to the Neural Network. As a result, two changes were made: setting a fixed sequence length for entries, and making sure the variables were understood as nominal, categorical datapoints. To accomplish the first, the only necessary step was to choose the desired sequence length and then to pad all sequences that were below that length with symbols to indicate blanks, and truncate all that were above. Based on the mean length of 242 Amino Acids for a sequence in the dataset, and the fact that this is roughly the size of larger domains, 250 amino acids was chosen as the final length, with the expectation that the neural network will have seen virtually all domains within these constraints and thus can apply it to longer sequences. This gave the input and ouput sets the shape [202634, 250].

In regard to the latter task, of encoding data in a machine-digestible format, each secondary structure element and residue was converted to a numerical equivalent one-hot encoding was used. One-hot encoding reshapes the data such that each position is replaced by a length n vector of boolean values, where n is the number of possible values, and the index of the value is set true. For sequences padded to length 250, this meant the vector would contain all zeros. As a result, our input dataset obtained the shape [202634, 250, 20] and our

Statistic	Value
Mean	142.084
Mode	99
Min	1
Max	3930
Total Sequences	202634
Sequences less than Mean	118892

Figure 2: Statistics for the master data set, including all non-redundant chains. Note: Redundancy has been cleared per-structure, not overall. If the same chain exists in 2 entries it is included.

output dataset obtained the shape [202634, 250, 8].

With the dataset fully prepared, it was important to create a subset of the data which the Neural Network had never encountered before to gauge widespread accuracy. The scikit train_test_split function was used to accomplish this due to the fact it allows developers to specify a numerical "seed" that will determine the way random sequences are selected, and thus can be used to populate the training and testing sets with the same sequences as used in this project. We utilized it with the seed value 2 and split 90% of the data into a set for training and 10% into a set for testing.

Set	Train Shape	Test Shape
Input	(182370, 250, 20)	(20264, 250, 20)
Output	(182370, 250, 8)	(20264, 250, 8)

Figure 3: The final shape of all sets derived.

3.2 Neural Network

Under the hood, Predict2Tensor utilizes a RNN (Recurrent Neural Network) built in Tensorflow[1] using the high-level API Keras to make predictions. Unlike older forms of RNNs however, the model utilizes a Bi-Directional LSTM (Long Short Term Memory) architecture. In the past, RNN models have relied on a technique called back-propagation, where neurons are uniquely weighted based on the amount of loss they incur individually. The issue with this method, however, is that limited numerical precision can incur issues where the weight assigned to a neuron will not be granular enough to make any difference. The effects of gradient descent are especially pronounced in the case of long sequences. LSTM based techniques have partially overcome these issues, doing so by decoupling the weight from the input and output as a value c and only allow additive changes. This allows the LSTM to contextually understand the input of the previous elements in the sequence when making a prediction. Building atop this concept, the model uses Bidirectional variants of these nodes which read the sequence in both directions, meaning that elements both prior and

subsequent are included in predictions. [14]

The final network utilizes 4 Bidirectional layers with decreasing density, and outputs are decoded on the final layer with a simple Dense layer, where each neuron corresponds to a single prediction. The gradient descent optimization algorithm chosen was ADAM, and the loss function used was categorical crossentropy. Parameters were tuned and determined empirically. The model was trained using Google's 2-8 Cloud TPUs as part of their Tensorflow Research Cloud program, which work optimally with batch size multiples of 64 between 64 and 1024. It was determined that a batch size of 512 and 10 training epochs yielded optimal results.

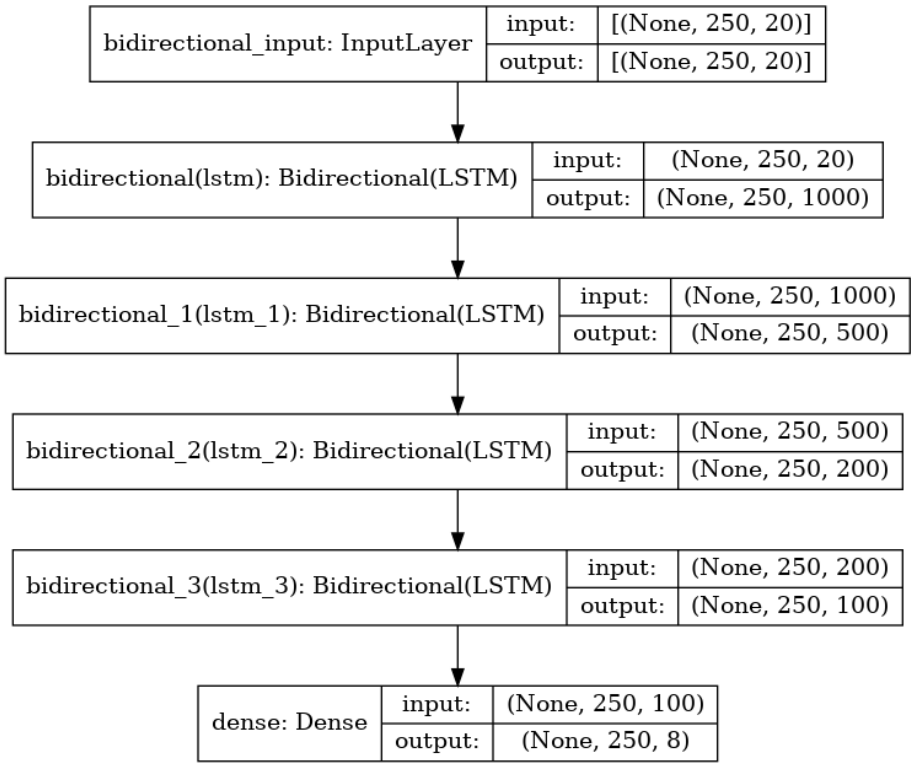

Figure 4: A diagram representing the flow of data through the network, including the size and shape of data passed at each layer.

3.3 Model Serving and Access

The Predict2Tensor model is accessible on the web from an intermediary server built with Node.JS [4], Webkit, MongoDB, and Redis. Redis is used as a database both for storing a queue managed by the Node.JS library Bull[10]. The results of predictions are then stored in a MongoDB collection. Dynamic

content such as results is loaded into pages by using the Lit (formerly LitElement) library to build dynamic components at-request.

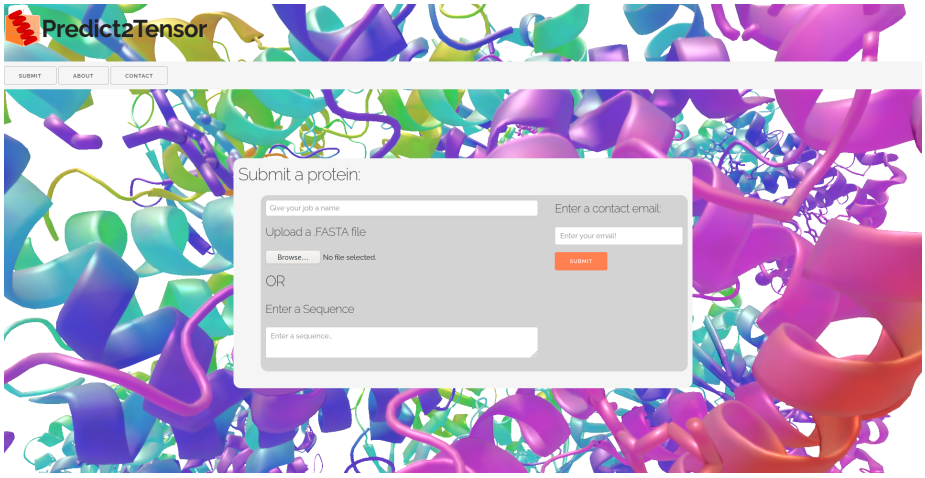

Figure 5: The landing page, where a user can submit a primary sequence query to be predicted.

The finalized model was served using Tensorflow Serving, a high-performance, flexible platform which allows straightforward deployment of Tensorflow models to run on CPUs, TPUs, or GPUs. The software is packaged and run as a Docker container, and HTTP GET requests can be made by packaging network input as a JSON query.

When the user submits a query, which includes a sequence, an ID, and an email address, it is tagged with the time of submission and then inserted into the Bull queue by the producer function. Bull automatically manages calling a consumer function which one hot encodes the submitted sequence, and submits it as a POST request to the Docker container. When a response is returned, which is in the format of the prediction strength for each potential structural element, two elements are extracted. The first of these is the prediction with the highest number, which is used as the categorical prediction. The second is a measure of confidence based on the differences in prediction strength, based on the following formula:

$$Confidence = \frac{\sum AP - MLP}{|AP| - 1} * 100\%$$

Where MLP is the most likely prediction that we just determined, and AP is the the set of all predictions. Since the neural network predicts 8 categories of secondary structure, its cardinality will always be 8.

The result is packaged with the original information from the query and entered into the MongoDB database as a JSON object along with a TTL (Time To Live) such that the database will automatically remove them after a week.

The user's supplied ID is used as a primary key, and thus are suffixed with an arbitrary 4 digit key to ensure they are unique.

Once a job is complete, the email address supplied is emailed a message informing them of completion of supplying a link to view the query. Queries can be accessed from *predict2tensor.com/results?query=jobID*, where jobID is the supplied identifier with its unique suffix. The results page presents the user with an alignment of primary and secondary structure, along with a position guide. Additionally, the aforementioned confidence values are displayed, by shading the background of the secondary structure elements accordingly. This results page is created using Webpack and lit (formerly known as litelement), a library for generating dynamic web components. [9]

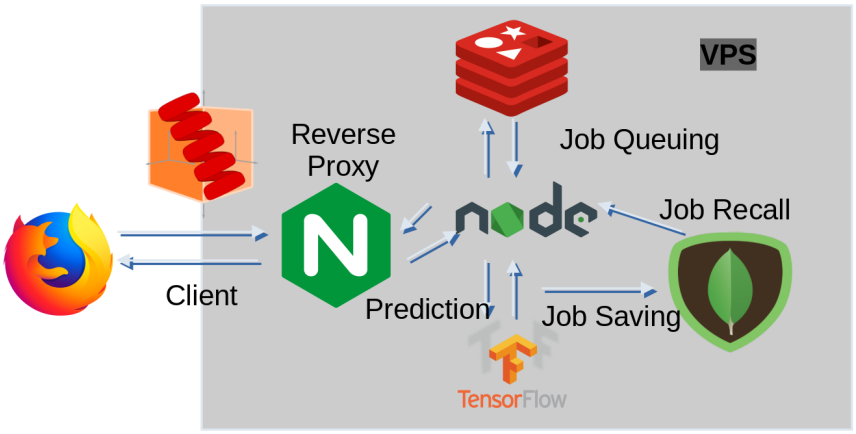

Figure 6: A chart representing the flow of queries through the server. The grey box represents processes running on the server.

4 Results

4.1 Accuracy

Overall, Predict2Tensor is shown to have an accuracy ranging from 92 percent to 80% based on structure length, beating out standard algorithms such as JPred and performing on the high end of cutting-edge methods. The overall average accuracy is 86%. The more common structural elements, such as beta sheets and alpha helices, as well as unstructured loop, have the highest accuracy, most likely because they have the largest supporting base. The neural network also has an unusual proclivity for identifying Pi helices, most likely due to the fact that the new DSSP algorithms have made themselves more sensitive and thus it has been included in our neural net. The lowest scoring element is a Beta bridge element, which is due to the fact that these elements are highly contingent on

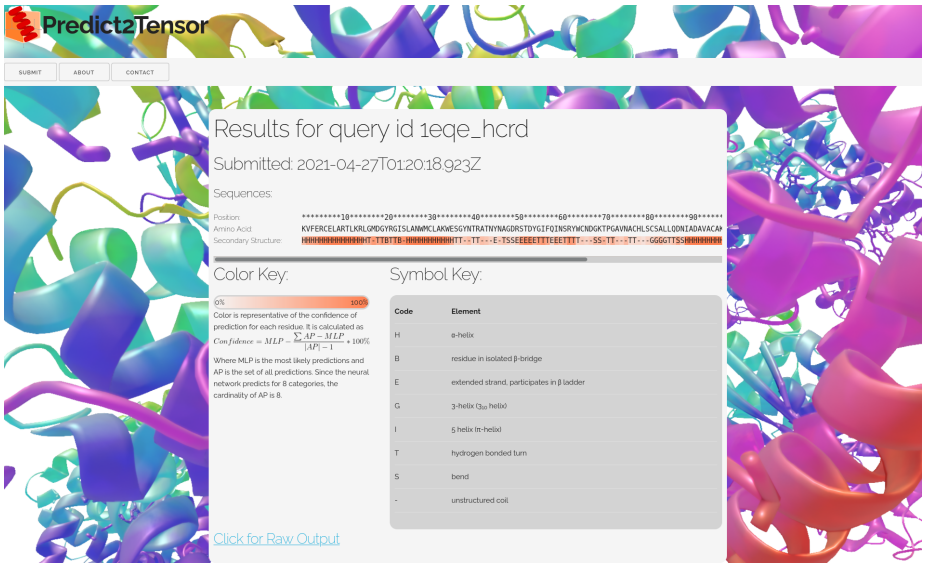

Figure 7: The results after querying the server to predict the secondary structure of Nitrile Hydratase, PDB 1AHJ.

3D proximity and thus cannot be identified easily from such limited information. One speculative approach to fixing this would to be implementing rudimentary fold information such as an HP-Algorithm fold.

Additionally, another variable that was considered when evaluating the neural network was how accurate it was at predicting discrete structural units. Thus, the Definition metric was defined, which assesses how many actual start and end sites are seen in the prediction, and how many predicted sites are seen in the actual sequence. This way, we understand how sensitive the network is to these regions, as well as how many erroneous structures are predicted. The overall Definition is (0.67, 0.74), though it varies based on length, as seen in figure 9. Generally, we see a trend of more predicted structures in the actual, presumably due to noise in predictions in the form of randomly inserted, short structures.

In order to understand the likelihood of different accuracy scores for a given prediction, two samples were taken: one from the overall set, and one from the test set. To observe the distribution of accuracies and confirm that there is no difference between sets, a histogram (Figure 10) was created. This confirmed the consistency of the sets, and shows that very few sequences are predicted below 70%. The cumulative percentages for every 5 percentages of accuracy were calculated, and a bar chart was created. (Figure 11) This revealed that the actual percentage of sequences with an accuracy about 70% is a striking 88%. Another impressive fact is that roughly 30% of predictions are above 95% accurate, though virtually all are short peptides, and/or are composed of helices

	precision	recall	f1-score	support
H	0.95	0.98	0.96	2669361
B	0.86	0.32	0.47	42005
E	0.82	0.92	0.87	772056
G	0.77	0.59	0.67	129324
I	0.79	0.65	0.71	22458
T	0.69	0.64	0.66	393866
S	0.68	0.49	0.57	313715
-	0.74	0.72	0.73	723215
accuracy			0.86	5066000
macro-avg		0.78	0.67	0.71	5066000
weighted-avg		0.85	0.86	0.85	5066000

Figure 8: The per-structure results for the final version of the network. Precision, as reported here by scikit, represents the percentage of true positives out of all positives, recall (also known as sensitivity) represents the percentage of true positives out of true positives and false negatives. The f1-score is the harmonic mean of the two.

	X<50	50<x<100	100<x<150	150<x <200	200<x<250	X=250
Support	3125	2290	2601	1998	3170	7080
Loss	0.249	0.252	0.319	0.418	0.378	0.573
Accuracy	0.916	0.913	0.889	0.855	0.872	0.802
Def. Avg.	0.622	0.725	0.795	0.812	0.893	0.855

Figure 9: The statistics based on the size of the sequence passed to the neural network. Def. Avg. is an abbreviation of definition average, which is the average of the actual to predicted definition and the predicted to actual definition.

and coils only.

As a final investigation into the accuracy of the network, two specific proteins were chosen from the 1000-sequence sample. The first of these was the A chain of PDB ID 3V90, which had an accuracy of 92.79%. The second was the A chain of PDB ID 3V90, with an accuracy of 41.99%. These two proteins are in the same range lengthwise, with the former being 260 amino acids and the latter being 334 amino acids, both exceeding the training limit but still being considered medium-size proteins. These proteins' primary sequence, DSSP-derived secondary structure, and the predictions of Predict2Tensor, JPred, and PSIPred were then aligned. Though the predictions can not be directly compared due to the fact that DSSP and by extension Predict2Tensor use eight-class prediction and PSIPred and JPred both use three-class prediction a few trends can be gleamed from the alignment of these predictions.

The high-accuracy prediction is that of the A chain of PDB ID 3V90, which is the structure a glycogenin transferase mutant. (See appendix Figure 13) All of the predictive methods seemed to quite accurately match the DSSP generation, especially in regard to the end of secondary structural elements. Where

Predict2Tensor exceeded, however, was in predicting details in the secondary structure such as turns and alternate helices, which JPred and PSIPred seemingly totally excluded from their predictions, considering them unstructured loops. Additionally, Predict2Tensor was able to identify the beginning of secondary structural elements more often than the other two methods.

The low-accuracy prediction is based on the B Chain of PDB ID 5HX2, an electron microscopy structure of the baseplate of the T4 Virus. (See appendix Figure 14). Between all prediction methods, all made wildly inaccurate and yet, conflicting predictions. Though all methods appear observant of the appearance of structures at certain positions, oftentimes prediction methods will disagree on which class the structure is, and in some cases will significantly mispredict the start and end sites for the structure. In the case of Predict2Tensor specifically, the network seems to overpredict many short structures inside coiled regions of other structures, a behavior which is also observed in JPred and PSIPred's predictions yet at a lower rate. We speculate 3 factors that could attribute to this poor prediction. Firstly, there is very few baseplate structures in PDB, and even those that exist have little homology with 5HX2, meaning the network likely has very little training on similar structures. Secondly, there is a high propensity of turns and bends in the structure. Finally, 5HX2 is a 9-peptide assembly, and it is unlikely the network has been trained on many sequences from structures composed of that many chains.

One additional note to consider in the comparison of these proteins is the validation of the confidence metric as a valuable visual indicator. In the case of the high-accuracy secondary structure prediction, the confidence was generally very high, around eighty percent for most positions. Additionally, many, but not all, of the sites of inaccurate predictions had lower confidence. On the other hand, the low-accuracy predictions were on average merely thirty percent confident, outside of a few correctly predicted structural elements which were around seventy percent confident.

4.2 Speed

One benefit Predict2Tensor has against established prediction methods is that it is incredibly computationally inexpensive. Though Predict2Tensor lacks the high load seen by JPred and PSIPred, it runs on a shared-CPU VPS with a single-core processor and 1GB of RAM alongside other software, and yet still returns jobs at a faster pace than either of . For comparison, JPred's documentation states that in the best case scenario of a small queue and a short sequence, queries can be expected to be completed in "less than a minute". Predict2Tensor, on the other hand, is able to complete small jobs in less than a second, given an empty queue. This is even more impressive when considering that a low-utilization conditions for JPred typically means around three hundred and fifty submissions, or one submission every four seconds on average. Unfortunately, due to the fact that JPred does not make the full-pipeline source code available, the exact quantitative difference cannot be assessed. PSIPred does in fact offer its source, and in later work the timings of the two will be di-

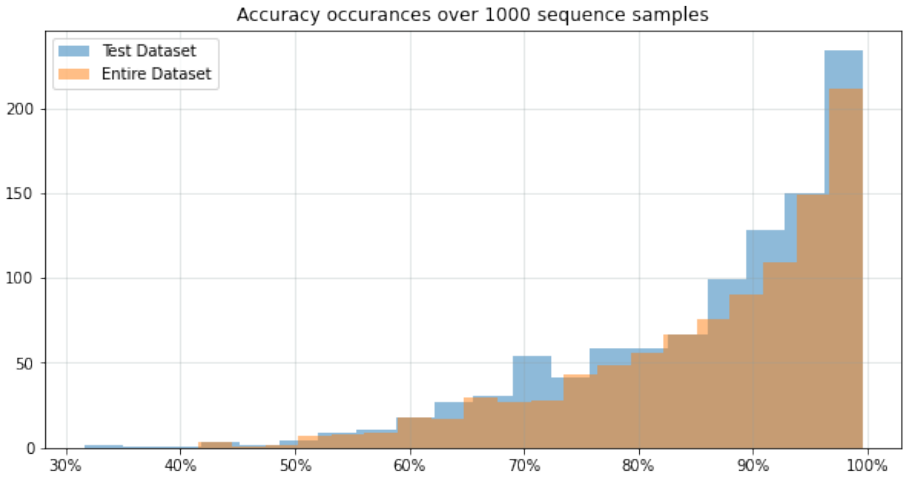

Figure 10: An assessment of the distribution of accuracies for a 1000 sequence sample pulled from the test set (blue) when compared to a sample from the entire dataset (orange).

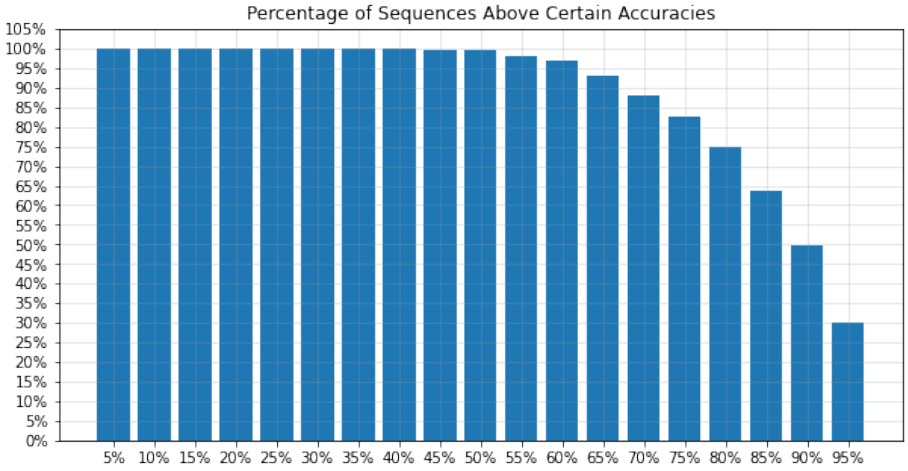

Figure 11: The cumulative percentage of sequences with accuracy above the given percentages.

rectly compared. As can be seen in Figure 12, shorter peptides can be predicted in around a quarter second, though for medium sized proteins one can expect prediction times in the range of eight hundred milliseconds to fifteen hundred milliseconds.

Outside of the statistically demonstrated speed, Predict2Tensor has an ad-

Chain	Length	1st Run	2st Run	3rd Run	4th Run	Average
1PPB_A	36	420	260	242	285	301.75
1AHJ_A	207	934	821	1005	777	884.25
3R1K_A	428	1747	1521	1578	1567	1603.25
6U5U_A	1887	5879	5882	9840	6891	7123

Figure 12: The time (in milliseconds) taken to transform a query, submit it to Tensorflow Serving, and receive and transform the response.

vantage in scalability, due to the technologies utilized. This scalability comes from a mix of horizontal scaling, where software is designed to utilize single machines with increases in computational resources, and vertical scaling technologies, where software is distributed across multiple containers or machines. Since Tensorflow Serving is distributed as a Docker image, it can be orchestrated using a platform such as Kubernetes. Orchestration platforms enable large volumes of Docker containers to be dynamically created and managed, and more importantly, enables load balancing. The Tensorflow Serving Docker image is designed with Kubernetes deployment in mind, and thus is well fit to such an approach.[12] The Bull queue and its functions are threaded and thus run in parallel, and due to the use of the Node.JS platform, the entire frontend can be trivially converted to docker images as well, further increasing scalability if need be. The two underlying DBMS platforms, Redis and MongoDB, are also highly tailored for scalability. As an in-memory key-value store, Redis is designed as a common data structure that can be shared between different processes, and is incredibly fast, but runs exclusively on a single core and thus does not scale vertically.[8] However, if needed, it also supports horizontal scaling in the form of sharding, a method by which the database is fragmented into multiple smaller database instances running on different instances. Another option could be to package discrete Redis instances with each frontend container. MongoDB has the benefit of inbuilt scaling in both directions, however in the case of vertical scaling, it suffers from a non-linear increase in computational cost. In regard to its horizontal scaling capabilities, MongoDB supports replication in addition to sharding. Replication gives each instance a complete copy of the database. Replication enables horizontal scaling with much faster read speeds, but due to the need to propagate writes to all instances, suffers from slow update speeds and occasional inconsistencies in data.[5]

5 Conclusions

Predict2Tensor is a fast, modern secondary structure prediction method based on recent advances in neural network technology. It has proven itself to be on par with recent, state of the art methods of secondary structure prediction while being simple enough to offer fast prediction times which can be scaled easily due to the use of modern cloud technologies. With the backing of modern

Neural Network frameworks and a large dataset, it's possible such an approach could even reach the sought-after limit of 88% average accuracy.[13] Regardless of whether or not it can easily achieve such a milestone, however, the high sensitivity and rich output offered by 8 class predictions at such a rapid prediction speed makes it useful not only for research, but to help accelerate compute-intensive tasks that integrate secondary structure as part of a larger pipeline, such as structural alignment and *de novo* folding methods.

5.1 Future Work

Due to the potential for further developments in both usability and accuracy, Predict2Tensor will continue to be in active development. The main focus is currently on increasing the accuracy. The first investigation into increasing the accuracy will be grid searching, in which a parameter space is generated based on the hyperparameters of the network, and the space is explored iteratively to determine the combination that yields the lowest loss and/or accuracy. Since grid searching requires a significant amount of training time, and Predict2Tensor was developed in 2 months, the parameters were determined empirically and there still could be a few percentage points gained. Another approach to increasing accuracy would be to potentially develop a model utilizing homology information. As can be observed in The project's github page, work has already developed on extracting the PDB file keywords from the dataset. In theory, these keywords could be used as a secondary input to the model, and could be integrated based on homology searching primary sequences and determining the keywords of homologous entries. It may also be wise to apply methods used by PSIPred and JPred such as multiple models or solvent accessibility calculations to Predict2Tensor as well. Other goals that are currently on the horizon for Predict2Tensor are designed around a better user experience, such as fully dynamic pages, automatic page reloading, and a documented, easily accessible POST API.
Once these advancements have been made, it would also be beneficial to compare Predict2Tensor in a manner closer to PSIPred and JPred, such as benchmarking it against the CB513 dataset, which has also been used for the two historical methods.

One can keep up with an updated list of goals and enhancements to Predict2Tensor on the Projects tab on its GitHub Page.

Predict2Tensor:predict2tensor.com
Source Code: https://github.com/robotheadache/predict2tensor

6 Bibliography

References

[1] *TensorFlow: A system for large-scale machine learning*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[2] James A. Cuff and Geoffrey J. Barton. "Application of multiple sequence alignment profiles to improve protein secondary structure prediction". In: *Proteins: Structure, Function, and Genetics* 40 (2000), pp. 502–511. DOI: 10.1002/1097-0134(20000815)40:3<502::aid-prot170>3.0.co;2-q. (Visited on 12/29/2020).

[3] Alexey Drozdetskiy et al. "JPred4: a protein secondary structure prediction server". In: *Nucleic Acids Research* 43 (Apr. 2015), W389–W394. DOI: 10.1093/nar/gkv332.

[4] Node.js Foundation. *Node.js*. Node.js, 2019. URL: https://nodejs.org/en/.

[5] MongoDB. Inc. *How to Scale MongoDB*. MongoDB. URL: https://www.mongodb.com/basics/scaling.

[6] R. P. Joosten et al. "A series of PDB related databases for everyday needs". In: *Nucleic Acids Research* 39 (Nov. 2010), pp. D411–D419. DOI: 10.1093/nar/gkq1105. (Visited on 05/07/2021).

[7] Wolfgang Kabsch and Christian Sander. "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features". In: *Biopolymers* 22 (Dec. 1983), pp. 2577–2637. DOI: 10.1002/bip.360221211. (Visited on 07/12/2019).

[8] Redis Labs. *Chapter 10: Scaling Redis—Chapter 10: Scaling Redis*. Redis Labs. URL: https://redislabs.com/ebook/part-3-next-steps/chapter-10-scaling-redis/ (visited on 05/07/2021).

[9] *Lit*. lit.dev. URL: https://lit.dev/ (visited on 05/07/2021).

[10] *OptimalBits/bull*. GitHub, May 2021. URL: https://github.com/OptimalBits/bull (visited on 05/07/2021).

[11] RCSB PDB. *File Download Services*. RCSB PDB, Apr. 2021. URL: https://www.rcsb.org/docs/programmatic-access/file-download-services#protocols-ftp-and-http (visited on 05/07/2021).

[12] *Use TensorFlow Serving with Kubernetes — TFX*. TensorFlow, Jan. 2021. URL: https://www.tensorflow.org/tfx/serving/serving_kubernetes (visited on 05/07/2021).

[13] Wafaa Wardah et al. "Protein secondary structure prediction using neural networks and deep learning: A review". In: *Computational Biology and Chemistry* 81 (Aug. 2019), pp. 1–8. DOI: 10.1016/j.compbiolchem.2019.107093. (Visited on 08/28/2019).

[14] Yong Yu et al. “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures”. In: *Neural Computation* 31 (July 2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199.

7 Appendix

PDB ID 3V90, Chain A	
Amino Acid Sequence	MTDQAFVTLTTNDAYAKGALVLGSSLKQHRTSRRLAVLTTPQVS
DSSP	-TTEEEEEEESSHHHHHHHHHHHHHHHHTT--SEEEEEE-TTS-
Predict2Tensor	--SSEEEEEESSHHHHHHHHHHHHHHHHTT--SEEEEEE-TTS-
Jpred	----EEEEEEE----HHHHHHHHHHHHH-----EEEEEE-----
PSIPred	---EEEEEEEE-HHHHHHHHHHHHHHHH------EEEEE-----
Amino Acid Sequence	DTMRKALEIVFDEVITVDILDSGDSAHLTLMKRPELGVMLTKLH
DSSP	HHHHHHHHHH-SEEEE|TTSTTSTTHHHHHHSGGGHHHHHHGG
Predict2Tensor	HHHHHHHHH|SEE----TTTTTTTTTHHHHH-GGHHHHHHGGG
Jpred	HHHHHHHHHH---EEEEE---------------HHHHHHHHHH
PSIPred	HHHHHHHHH---EEEEEE------HHHHHHH----HHHHEEHHH
Amino Acid Sequence	CWSLTQYSKCVFMDADTLVLANIDDLFEREELSAAPDPGWPDCF
DSSP	GGG-TTSSEEEEE-TTEEE-S|GGGGGS-SSEEEE-SSSTTSE
Predict2Tensor	GGG-TTSSEEEEE-TTEEE-S|GGGGGSSSSEEEE-SSSTTSE
Jpred	--------EEEEE-----EE----------------------
PSIPred	HHH-----EEEE----EEEE--HHHHH----HH----------
Amino Acid Sequence	NSGVFVYQPSVETYNQLLHVASEQGSFDGGDQGLLNTFFNSWAT
DSSP	EEEEEEE---HHHHHHHHHHHHHH--TTSSHHHHHHHHTTTTTT
Predict2Tensor	EEEEEEE---HHHHHHHHHHHHHH--TTSSHHHHHHHHTTTTTT
Jpred	---EEEEE--HHHHHHHHHHHH---------HHHHHHHH-----
PSIPred	-EEEEEEE--HHHHHHHHHHHHHH--------HHHHHH------
Amino Acid Sequence	TDIRKHLPFIYNLSSISIYSYLPAFKAFGANAKVVHFLGQTKPW
DSSP	S-GGGB|GGGSEETTHHHHTHHHHHHGGG-SEEE--SSS-GG
Predict2Tensor	S-GGG---GGGSEETHHHHHHHHHHHHHGGG--EEE--SSS-GG
Jpred	----------------H---HHHHH------EEEEEE------
PSIPred	--------HHH----HHHHH-HHHHHHHHH--EEEEE-------
Amino Acid Sequence	NYTYDTKTKSVRSMTHPQFLNVWWDIFTTSVVPLLQQFGL
DSSP	GS-EETTTTEE-----THHHHHHHHHIIIIIHHHHTTTT-
Predict2Tensor	G-EEETTTTEE-----HHHHHHHHHHIIIIIHHHHHHT--
Jpred	---------------HHHHHHHHHHHHHHHHHHHHHHH---
PSIPred	---------------HHHHHHHHHHHHH--HHHHHHH--

Figure 13: A comparison of predictions for a high-accuracy prediction by Predict2Tensor against other methods.

PDBID 5HX2, Chain B	
Amino Acid Sequence	MNDSSVIYRAIVTSKFRTEKMLNFYNSIGSGPDKNTIFITFGRS
DSSP	-TT-SS--EE--TTHHHHHHHHHHHHS--SSSSS--EEE-----
Predict2Tensor	-TTT-EEEEEEEEEE---HHHHHHHH---SSSS--EEEEEEE--
Jpred	----EEEEEEEH--HHHHHHHHHHHHH-------EEEEEEEE--
PSIPred	----HHHHHHHH-HHHHHHHHHHHHHHH-------EEEEEEE--
Amino Acid Sequence	EPWSSNENEVGFAPPYPTDSVLGVTDMWTHMMGTVKVLPSMLDA
DSSP	S|SSS-SSSSS------SSHHHHHHHHHH---B-EE-TTT-EE
Predict2Tensor	----SSS----------S-EE-HHHHHHHIIIIEEEEEEEEEE
Jpred	----------------------HHHHHHHHHHHE--------E
PSIPred	-------------------HHHHHHHHHHHHHEEE--HHH--E
Amino Acid Sequence	VIPRRDWGDTRYPDPYTFRINDIVVCNSAPYNATESGAGWLVYR
DSSP	----EESS-SSSS-SS---SS-BEE-SSSTTT--STTSS-----
Predict2Tensor	EE----TT-SS----T----T-EEEEEE-----TTTS-EEEEEE
Jpred	EEE------------------EEE----------HHHHHHEEE
PSIPred	EEE------------------EEE------EEEE----EEEE
Amino Acid Sequence	CLDVPDTGMCSIASLTDKDECLKLGGKWTPSARSMTPPEGRGDA
DSSP	EEEE|SS----SSS--STTTTTTS------S--------S--S
Predict2Tensor	EE---TT--------S-HHHHHHHT------GGT-------T--
Jpred	EEEE----EEEEEE-----E----------------------
PSIPred	EEE---------------HH----------------------
Amino Acid Sequence	EGTIEPGDGYVWEYLFEIPPDVSINRCTNEYIVVPWPEELKEDP
DSSP	SS----SSS-EEE------HHHHHHH--SS---B--TTTTTT-T
Predict2Tensor	TS-E-TT|EEEEEEEEE-HHHHHHHHTSSSEEE---HHHHH-T
Jpred	---------EEEEEEE-----HHEEE----EEEEE-----HHHH
PSIPred	---------EEEEEEEEE-HHHHH-------EE----------
Amino Acid Sequence	TRWGYEDNLTWQQDDFGLIYRVKANTIRFKAYLDSVYFPEAALP
DSSP	TTS|SS-TTTGGG-S--TTTTT---EEEEEEE-TTT-TTT-ST
Predict2Tensor	TT----TT-HHH--T--EEEEE-EEEEEEEEEEEEE---TTT-T
Jpred	HH------EEEE----EEEEEEEEEEEEEHHH-----------
PSIPred	---EEE--EEEEE-----EEEEE-EEEEEEEEE----------
Amino Acid Sequence	GNKGFRQISIITNPLEAKAHPNDPNVKAEKDYYDPEDLMRHSGE
DSSP	T----S---EEE--B---SSSSS---B--SSSB-STTSTTT-S-
Predict2Tensor	T-S-EEEEEEEE--------TS-TT---------TGGG----S-
Jpred	-----EEEEEE--HHHH---------E---------------E
PSIPred	-----EEEEEEE-----------------EE-HHHH----EE
Amino Acid Sequence	MIYMENRPPIIMAMDQTEEINILFTF
DSSP	-------S-----TT--EEEEEEE--
Predict2Tensor	EEEEES---EEE------EEEEEEE-
Jpred	EEEE-----HHH----HHHEEEEE--
PSIPred	EEEEE----------EEEEEEEEE-

Figure 14: A comparison of predictions for a low prediction by Predict2Tensor against other methods.