Expanding the Vocabulary of a Protein: Application of Subword Algorithms to Protein Sequence Modelling

Expanding the Vocabulary of a Protein: Application of Subword Algorithms to Protein Sequence Modelling

Mark Lennox[1], Neil Robertson[2], Barry Devereux[3]

***Abstract*— Deep learning has proven to be a useful tool for modelling protein properties. However, given the variability in the length of proteins, it can be difficult to summarise the sequence of amino acids effectively. In many cases, as a result of using fixed-length representations, information about long proteins can be lost through truncation, or model training can be slow due to the use of excessive padding. In this work, we aim to overcome these problems by expanding upon the original vocabulary used to represent the protein sequence. To this end, we utilise two prominent subword algorithms that have been previously used to reach state-of-the-art results in various Natural Language Processing tasks. The algorithms are used to encode the original protein sequence into a set of subsequences before they are analysed by a Doc2Vec model. The pre-trained encodings produced by each algorithm are tested on a variety of downstream tasks: four protein property prediction tasks (plasma membrane localization, thermostability, peak absorption wavelength, enantioselectivity) as well as drug-target affinity prediction tasks over two datasets. Our results significantly improve on the state-of-the-art for these tasks, demonstrating the benefits of using subword compression algorithms for modelling proteins.**

I. Introduction

As is the case in many other information processing domains, deep learning frameworks are emerging as powerful and highly effective tool in Bioinformatics, including in the area of predicting important protein properties from protein sequence information. Key to the success of deep learning solutions is their ability to extract complex task-relevant features from raw input representations. Before a deep neural network can analyse a protein sequence, it is typically transformed into one of two formats. The first option is to use a one-hot encoding of the sequence of amino acids [19, 10, 12], which involves replacing each character of the sequence with a one-hot vector. This is computationally feasible, given the limited size of the amino-acid vocabulary. Another option is to variable-encode the sequence of amino acids [18, 15, 21], where each amino acid is encoded by a unique integer. A trainable embedding layer within the deep neural network is then used to convert the sequence of integers into a set of trainable vectors, each representing an amino acid. One drawback of this approach arises from the limited size of the amino acid vocabulary. Given the small number of naturally occurring amino acids, this approach restricts the amount of syntactical information retrained by the embedding layer, which can result in the embedding layer over-fitting to specific patterns within the protein sequence.

*This work was not supported by any organization

[1]EEECS, Queen's University Belfast, mlennox05@qub.ac.uk
[2]EEECS, Queen's University Belfast, n.robertson@qub.ac.uk
[3]EEECS, Queen's University Belfast, b.devereux@qub.ac.uk

Modelling proteins in this way is equivalent to analysing a sentence only at the character level. However, representing the patterns within the sequence at such a low level can impact the amount of information that a deep neural network can effectively learn about the relationships between amino acids. Such limitations that begin in the embedding layer may adversely effect the performance of the network at a later stage. In this paper, we present an approach to mitigating these issues by first using subword algorithms to re-represent the protein sequence over a richer vocabulary. The algorithms we investigate each encode the proteins into a set of frequently occurring sub-sequences (k-mers), thereby reducing the overall sequence lengths (and the variability in sequence lengths), and effectively compressing the amount of data analysed by the deep neural network. In addition, this subword encoding process minimises the overall time and computational cost required to train the models.

A majority of the deep learning techniques adopted for analysing proteins have come from the field of Natural Language Processing (NLP). A common problem in NLP is accounting for and representing out-of-vocabulary (OOV) words, for example when translating a sequence of words from one language into another via neural machine translation (NMT) [1, 20, 34, 31]. One option to avoid the OOV problem is to use a character-level encoding. However, in the context of protein sequence modelling, modelling at this low level of representation may lead to a failure to capture important long-distance structural information within the protein sequence. Another approach to alleviating this issue has been to represent the protein sequence as a series of k-mers, which are sub-sequences of a fixed length of k amino-acids [36, 15, 35]. This approach has the desired effect of reducing the length of protein representations while also increasing the vocabulary size used for encoding the sequence. However, this simple approach does not take into account the relative frequency of occurrence of different k-mers: some k-mers may appear very infrequently within protein representations. Because they occur so infrequently in the training data, there is relatively little information for the deep learning model to learn from, which can lead to poor embedding representations for use in downstream tasks.

An additional encoding strategy has become popular within NLP that offers the benefits of encoding a protein at both an amino-acid and k-mer level, known as subword encoding. During the data processing stage, this encoding method breaks the original sequence into a set of sub-sequences of varying length [27, 4, 28, 34]. The two most popular methods currently used in the field of deep learning

are byte-pair (BPE), and unigram encoding, both of which will be considered in this investigation. These subword algorithms have become popular as a means of reducing the number of out-of-vocabulary words analysed by the task-specific neural networks. As a part of the data processing pipeline, these algorithms can convert the original sentence into a set of unique commonly occurring subsequences. In the context of protein sequence modelling, this involves a re-coding of the original amino acid sequence as a shorter sequence that uses a much larger vocabulary of symbols that represent commonly-occurring amino acid sub-sequences.

In addition to n-grams, another common strategy to counteract the limitations of one-hot and variable encoding is to use a particular deep learning layer known as a convolution layer [18, 14, 33]. A one-dimensional convolutional layer can be seen as a motif collector that combines the specific encodings for each amino acid to form a set of feature vectors. There is still no standard practice with regards to using convolutional layers for analysing proteins as the hyper-parameters vary for each use case. Another drawback to using the CNN architecture is that it is only capable of collecting common motifs within the sequence and is incapable of actually modelling the long-range dependencies within the data.

In this work we focus on comparing our approach based on subword encodings to previous work using n-grams. In the experiments described below we have opted for using a Doc2Vec model [16] to test the performance of each encoding style. The embeddings produced by each Doc2Vec model will form the input to Gaussian process (GP) regression models [25]. This will allow for a fair comparison to the work carried out by Yang et al. that was based on a tri-gram approach [36]. We hypothesise that the encodings provided by each subword algorithm will improve the overall performance of a deep learning model, as these algorithms will be able to compress more information into each window being modelled by the Doc2Vec algorithm.

II. Materials and methods

Analysing the primary structure of a protein can be viewed as analogous to much of the work currently being carried out in the field of NLP which relies on learning the linguistic structure of sentences. Recent work in NLP has demonstrated the efficacy of *pre-training* — in this technique, a model is initially trained on a very large corpus of unlabelled text, before being fine-tuned using labelled data for some specific task. Conceptually, pre-training allows the model to learn the statistical regularities of the language (i.e. the meaning of words and the grammatical relationships that can exist between them), whilst fine-turning optimises the network for a particular task (e.g., identifying the emotional sentiment of a sentence). Pre-training such language models on large scale unlabelled data typically requires a considerable amount of computational power. However, the encodings produced are general enough for a wide variety of downstream tasks [17, 3, 30].

Pre-training is now standard within NLP and has resulted in networks such as ELMo [22], GPT [23], GPT-2 [24], BERT [7], and XLNet [37] achieving state-of-the-art results for language modelling and subsequent downstream tasks. In nearly all cases these models use a subword algorithm to first re-encode the original text. This transformation allows the system to separate rare words within the vocabulary into more common subwords; for example, the word “cars” might be split into the tokens “car” and “s”. By breaking up rare words within a corpus, it simplifies the modelling process by allowing the network to combine these subword representations to represent words, instead of using the original character sequence of the words.

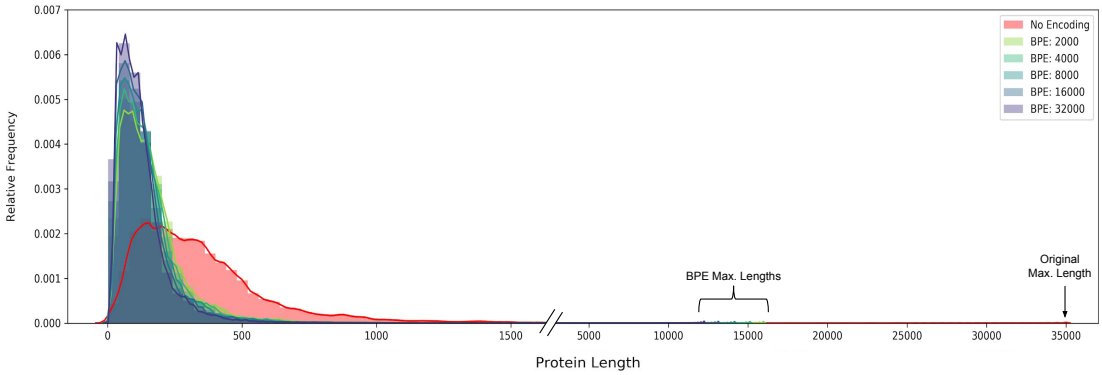
For this investigation, we pre-train a set of Doc2Vec models on the most recent version of the UniProt database of verified proteins [5]. The current version of the database provides $\sim$500,000 known proteins that will form the corpus used to both optimize the subword algorithms (byte-pair encoding & unigram encoding) and train the Doc2Vec models.

A. Byte-Pair Encoding

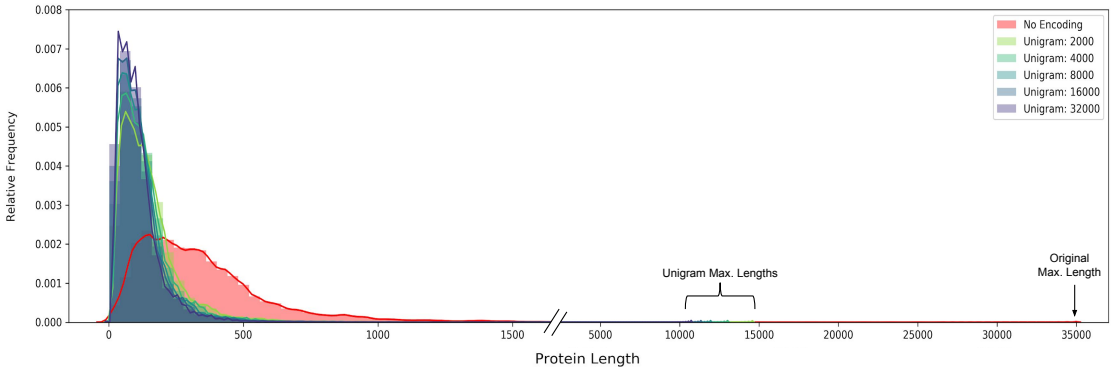
Byte-Pair-Encoding (BPE) is the most popular subword segmentation algorithm currently being utilised within the field of deep learning [9, 28, 27]. It is based on a dictionary-style encoder that ultimately minimises the total number of symbols required to reproduce the original sequence. For a protein, the algorithm partitions the sequence of amino acids into a set of frequently occurring subsequences (i.e. k-mers). Subsequences are derived interatively by combining the most frequent adjacent pairs of subsequences into a new symbol until a maximum vocabulary size is reached. Once optimised, the new vocabulary is used to break any protein into a sequence of k-mers, thereby reducing the actual length of the token sequence required for analysis (Fig. 1a). One clear disadvantage of this algorithm is that it is purely based on the frequency of adjacent pairs of symbols, which may affect how the algorithm breaks a protein up into k-mers. For example, if a specific motif is poorly represented within the set of training proteins, then the algorithm will struggle to accurately split a motif into its constituent k-mers.

B. Unigram Encoding

Another approach to subword segmentation is based on optimising the entropy of a given encoded sequence. The unigram encoding algorithm has only been recently developed but has become popular within the field of NLP for its ability to generate multiple subword representations [13]. Using multiple representations has the added benefit of allowing deep learning algorithms to generalise better to these encodings, compared with just using a single representation as in the case of BPE. In the context of encoding proteins, this algorithm treats each subsequence of amino acids independently to its neighbours. At each iteration, the vocabulary is preset by the algorithm so that each protein can be broken up into a set of k-mers reducing its overall length (Fig. 1b). The expected probability of these encodings is then maximized using a Viterbi algorithm [32]. This process



(a) Byte-Pair Encoding.



(b) Unigram Encoding.

Fig. 1: The distribution of protein lengths after they are encoded by different subword algorithms.

continues as the algorithm iteratively reduces the vocabulary size until the maximum specified size is reached.

C. Tasks

The four downstream tasks included in this paper contain a diverse range of proteins with each task measuring different properties. These tasks were previously used to evaluate protein embeddings by Yang et al. [36], allowing us to compare our subword encoding strategies to the tri-gram encodings considered in that work. In this section, we will only briefly describe the datasets used (for more detail on the data and the collection methods used in each of the task datasets, please see citations). The peak absorption wavelength dataset consists of Gloeobacter violaceus rhodopsin (GR) parent and 80 sequences that include 1–5 mutations [8]. The enantioselectivity dataset consists of epoxide hydrolase (EH) parent and 151 sequences that include 1–8 mutations [38]. The plasma membrane localization dataset comprises a total of 248 sequences [2]. The Thermostability (T50) dataset includes a total of 261 sequences [26]. In summary, these four tasks include a variety of measured protein properties as they include libraries constructed from both recombination and site-directed mutagenesis. The four tasks are summarized in Table 1 of [36].

In addition to the four downstream tasks considered by Yang et al. [36], the application of subword algorithms to a set of drug-target affinity tasks will also be investigated. These datasets were previously used to evaluate drug-target affinity prediction using a convolutional neural network (CNN) architecture by Öztürk et al. [21], allowing us to compare our subword encoding + Doc2Vec framework to this method. The Davis dataset was a study into the interactions between 442 unique proteins (taken from the kinase protein family), and 68 individual drugs with the affinity of each pair being measured as a dissociation constant (K_d) value [6]. The Kiba dataset included 229 unique proteins and 2111 individual drugs [29]. The affinity values used in this task were based on a Kiba approach described by Tang et al. [29]. This diversity across all tasks allow us to properly evaluate the generality of the embedded representations. The Davis and Kiba datasets are summarized in Table 1 of [21].

D. Modelling Scheme

The modelling scheme used for this investigation is outlined in Figure 2. It is split up into three key stages; in the initial stage, a subword algorithm (e.g. unigram or BPE) is optimized on the ∼500,000 known proteins collected from the UniProt database. Once the subword algorithm has been optimized, a Doc2Vec model is then pre-trained using the encodings produced by the subword algorithm (as opposed

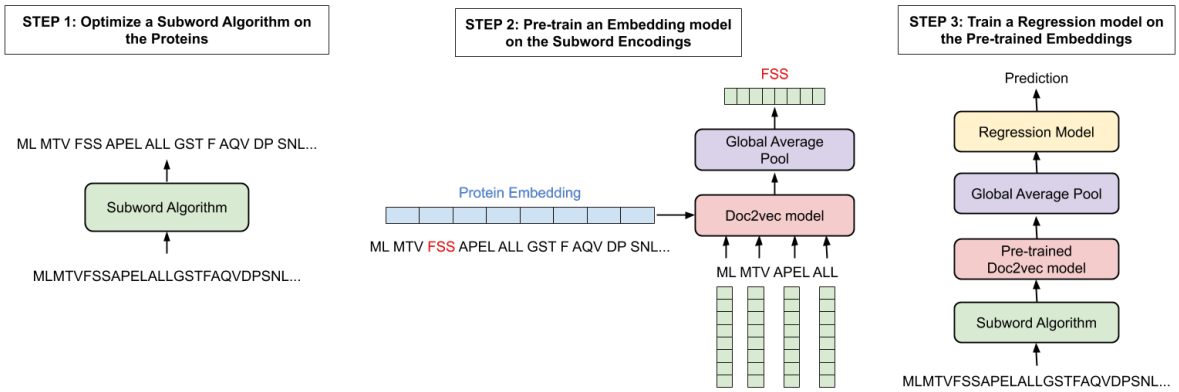


Fig. 2: An outline of the modelling scheme.

to k-mer encodings [36]). Finally, once the Doc2Vec model has completed its training cycle on the encoded corpus, it can then be used to produce a vector representation of any protein of interest. The encodings produced by the pre-trained Doc2Vec model can then be used by a suitable regression model to provide predictions for a protein property (such as drug affinity). For a fair comparison to the work carried out by Yang et al. [36], we have opted to also use Gaussian process (GP) regression models [25] based on Matérn kernels with $\nu = 5/2$, which was fixed for all downstream tasks. Likewise, for an unbiased comparison to Öztürk et al.'s [21] results for predicting drug-target affinity, we also use a set of fully connected layers to combine and model the embedded representations of both the drug and the target protein for these tasks.

III. Results and Discussion

A. Protein property tasks

We began our analysis by comparing the results of a set of Gaussian Process regression models for the four downstream tasks from [36]. In each of the four tasks, the subword equivalent Doc2Vec models were a considerable improvement over the original tri-gram [36] and character-encoded baselines (Table I). Figure 3 depicts the similarity structure of the best performing model for each task (bold entries in Table I), and how the similarity structure relates to both the property being predicted in each task and the number of mutations. To produce these embeddings, a unigram algorithm ($v = 4000$) along with a Doc2Vec model ($w = 3$) was chosen for absorption. For enantioselectivity, a byte-pair algorithm ($v = 16000$), and a Doc2Vec model ($w = 3$) is presented. For localization, a unigram algorithm ($v = 2000$), and a Doc2Vec model ($w = 5$) is presented. For T50, a byte-pair algorithm ($v = 8000$), and a Doc2Vec model ($w = 3$) was used, where v denotes the vocabulary size of the subword algorithm and w denotes the window size of the Doc2Vec model. Following Yang et al. [36], the first two columns depict a t-SNE plot for the best embedding solution (coloured by task property and number of mutations). The triangles indicate the parent proteins for each task. The third column depicts a correlation matrix for the protein pair (with rows/columns of the heatmap ordered by a hierarchical clustering solution).

The t-SNE plots for the peak absorption wavelength task show a clear separation between proteins with high and low absorption values. The correlation matrix reveals two clear clusters within the dataset (i.e. low and high). There is a relationship between the number of mutations a protein has and its peak absorption wavelength; this relationship is captured by the pre-trained Doc2Vec models which successfully cluster similar proteins together and show the link between mutation and absorption. Both subword encodings revealed noticeable improvements over the tri-gram technique used by Yang et al. [36], with the unigram encoding scheme outperforming BPE (Table I).

The enantioselectivity task was the only task where BPE and a more extensive vocabulary was optimal. Again, as shown in Table I, both subword algorithms perform similarly. Just as in the absorption task, we see from the t-SNE plot and the correlation matrix that the Doc2Vec encodings again contain information that links the mutation to a protein's enantioselectivity. As in the absorption task, there is a clear pattern to how the Doc2Vec model encodes the proteins that captures the number of mutations and enantioselectivity values, and the relationship between them.

Similar results were found for the plasma membrane localization task. Both subword encodings again had examples that vastly improved upon the previous tri-gram approach (Table I). However, when considering the t-SNE and correlation plots, we see that the Doc2Vec encodings cluster the proteins on the basis of their mutations, as there are three clear clusters for each of the three parent proteins (indicated by the triangles in the t-SNE plots). Within each of these clusters, the actual task property of membrane localization was moderately represented by the Doc2Vec encodings. For each of the three groups, we see that the encodings lack enough detail to properly separate the proteins from high and low membrane localization values. However, it should be noted that from the Doc2Vec encodings, Figure 3, we can easily identify which of the parent proteins is more

TABLE I: Results for the protein downstream tasks.

Sudword Encoding	Vocabulary Size	Absorption	Enantioselectivity	Localization	T50
BPE	2000	23.83	10.38	0.66	2.70
	4000	20.80	9.76	0.67	3.01
	8000	18.46	6.72	0.75	2.75
	16000	20.64	**6.08**	0.73	2.76
	32000	24.27	7.03	0.67	2.80
Unigram	2000	26.41	6.77	**0.65**	2.98
	4000	**18.09**	6.90	0.76	2.80
	8000	20.92	8.58	0.86	**2.59**
	16000	24.05	7.07	0.77	3.33
	32000	21.98	9.53	0.76	2.96
Tri-gram [36]	8000	23.30	9.14	0.73	2.91
Character	20	46.08	12.55	0.81	4.32

Notes: Mean Absolute Error (*MAE*) between the actual test values and the predicted test values.

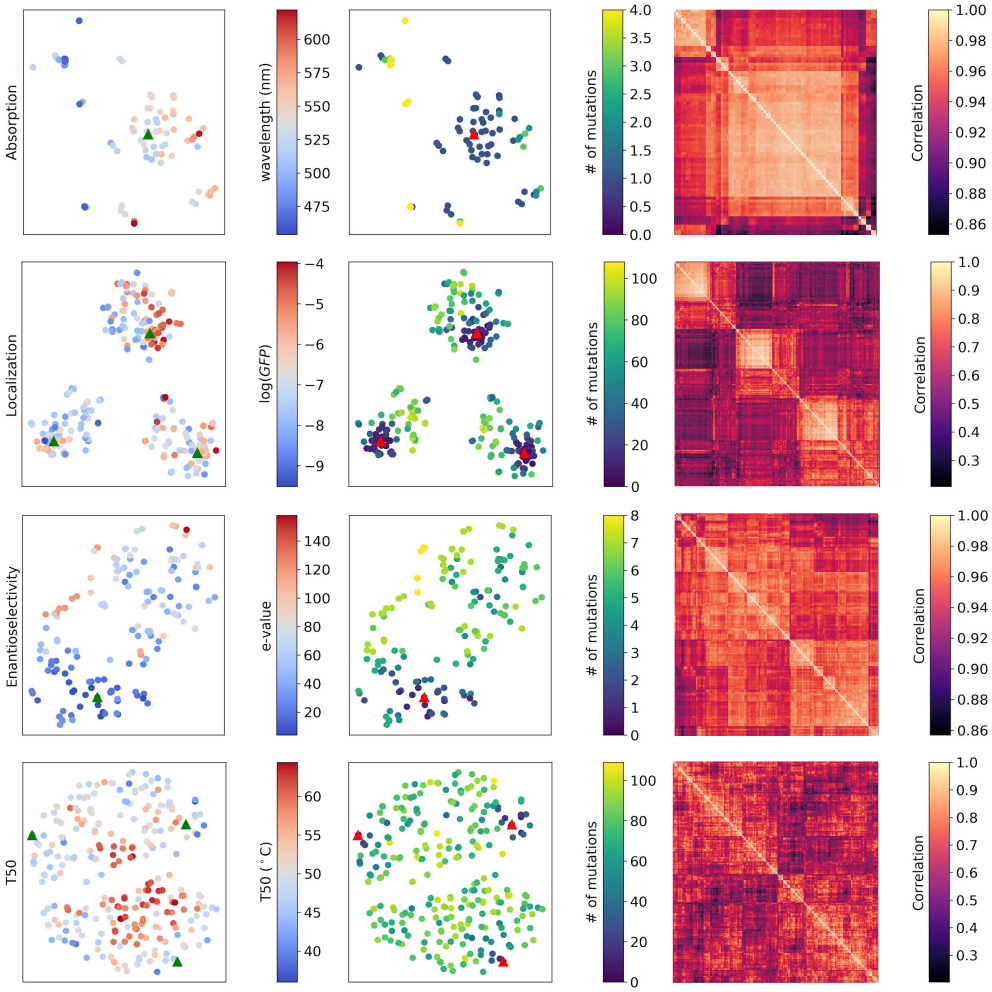


Fig. 3: t-SNE and correlation plots visualising the best encodings for each downstream task.

susceptible to having a higher localization value.

For the Thermostability (T50) task, we again see a performance boost for using a subword encoding over the baselines (Table I). In Figure 3, we observe the opposite effect to that of the localization encodings. In this task, the Doc2Vec encodings cluster proteins that share similar T50 values as opposed to purely clustering the proteins based on the number of mutations. From the t-SNE plots, one can see that that one particular protein is clustered closer to the proteins that share a high thermostability property. In contrast, the other two-parent proteins are clustered further away from this site. The overall lack of clustering observed from the encodings may be due to the Doc2Vec model being unable to capture sequential information or being able to model long-

TABLE II: Results for the drug-protein interaction tasks.

Dataset	Sudword Encoding	Model	MSE	r^2	$C.I.$
Davis	BPE	Doc2Vec	**0.210**	**0.715**	**0.903**
Davis	Unigram	Doc2Vec	0.213	0.705	**0.903**
Davis	Character	CNN [21]	0.261	0.630	0.878
Kiba	BPE	Doc2Vec	**0.190**	**0.678**	**0.874**
Kiba	Unigram	Doc2Vec	0.192	**0.678**	**0.874**
Kiba	Character	CNN [21]	0.194	0.673	0.863

Notes: Mean Absolute Error (*MAE*), r^2 regression score (r^2), and Concordance Index ($C.I.$) between the actual test values and the predicted test values.

range dependencies within the proteins.

Nevertheless, using either subword encoding schemes provides a definite improvement over the baseline encoding approaches. From Figure 3, we can discern how the number of mutations has a negative association with both the absorption and localization properties of a protein, with the opposite pattern for the enantioselectivity and T50 values.

To further improve the unigram results of all four tasks, we could consider the regularisation method by Kudo et al. [13] for open-vocabulary NMT modelling. Such a technique may prove useful during the pre-training stage of the protein modelling as it would produce multiple subsequence segmentations of each protein in the original pre-training corpus, which could lead to the more reliable and robust representation of the proteins.

B. Drug-Target affinity tasks

For the drug-target interaction tasks, we used the same datasets used in previous work by Öztürk et al. [21]. We used a 95:5 validation split of the training dataset to build our corresponding interaction multi-layer perceptron (MLP). This deep neural network was optimised using the ADAM [11] optimiser for a total of 100 epochs. To determine the best hyperparameters for the Doc2Vec model for both the drug and target data, we performed a simple preliminary test. A linear regression model was used to determine the optimal Doc2Vec hyperparameters for encoding both the drug and protein target. The MLP model was trained using the encodings produced by these optimal pre-trained Doc2vec models. Again, the results of the drug-target affinity task reveal the advantages of using both a subword encoding algorithm and a pre-trained model, as shown in Table II. In both the Davis and the Kiba datasets, these pre-trained subword based Doc2vec models produced far better results than the previous CNN-based state-of-the-art model [21], with only marginal differences in the performance between the byte-pair and unigram encoded models.

These results again demonstrate the performance boost from using subword encoding and pre-training on a larger corpus. It should also be noted that the Doc2Vec results were also a remarkable improvement over a far more sophisticated baseline (i.e. a three-layer CNN model), which would contain far more parameters than that of our relatively simple Doc2Vec model.

IV. Conclusion

The results of this investigation show the clear advantages of applying a subword algorithm to first encode a protein. Not only do these algorithms increase the size of the vocabulary, and compress the length of the protein, but they also improve the embeddings of a pre-trained model in every downstream task we tested, surpassing the state-of-the-art for these tasks and datasets. Whilst the correlation maps display strong and meaningful clustering within the data for each task, the pre-trained representations still do not reliably predict certain measured properties, such as localization and thermostability. Nevertheless, these promising results introduce the question of whether subword algorithms can further improve the performance when combined with more sophisticated deep learning algorithms, such as convolutional neural networks or long-short term memory networks.

We have shown that by first using a subword algorithm to encode the protein before performing pre-training, one can enhance the amount of information gleaned from the raw data. These techniques for efficient encoding of proteins allow us to take full advantage of the rise in the number of known protein sequences, while at the same time also reducing the cost and time required to model properties of interest accurately.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Claire N Bedbrook et al. "Structure-guided SCHEMA recombination generates diverse chimeric channelrhodopsins". In: *Proceedings of the National Academy of Sciences* 114.13 (2017), E2624–E2633.

[3] Ankush Chatterjee et al. "SemEval-2019 task 3: EmoContext contextual emotion detection in text". In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. 2019, pp. 39–48.

[4] Rohan Chitnis and John DeNero. "Variable-length word encodings for neural translation models". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 2088–2093.

[5] UniProt Consortium. "UniProt: a hub for protein information". In: *Nucleic acids research* 43.D1 (2014), pp. D204–D212.

[6] Mindy I Davis et al. "Comprehensive analysis of kinase inhibitor selectivity". In: *Nature biotechnology* 29.11 (2011), p. 1046.

[7] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[8] Martin KM Engqvist et al. "Directed evolution of Gloeobacter violaceus rhodopsin spectral properties". In: *Journal of molecular biology* 427.1 (2015), pp. 205–220.

[9] Philip Gage. “A new algorithm for data compression”. In: *The C Users Journal* 12.2 (1994), pp. 23–38.

[10] Fei He et al. “A multimodal deep architecture for large-scale protein ubiquitylation site prediction”. In: *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2017, pp. 108–113.

[11] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[12] Michael Schantz Klausen et al. “NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning”. In: *Proteins: Structure, Function, and Bioinformatics* 87.6 (2019), pp. 520–527.

[13] Taku Kudo. “Subword regularization: Improving neural network translation models with multiple subword candidates”. In: *arXiv preprint arXiv:1804.10959* (2018).

[14] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. “DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier”. In: *Bioinformatics* 34.4 (2017), pp. 660–668.

[15] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. “DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier”. In: *Bioinformatics* 34.4 (2018), pp. 660–668.

[16] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: *International conference on machine learning*. 2014, pp. 1188–1196.

[17] Jinhyuk Lee et al. “Biobert: pre-trained biomedical language representation model for biomedical text mining”. In: *arXiv preprint arXiv:1901.08746* (2019).

[18] Zhen Li and Yizhou Yu. “Protein secondary structure prediction using cascaded convolutional and recurrent neural networks”. In: *arXiv preprint arXiv:1604.07176* (2016).

[19] Fenglin Luo et al. “DeepPhos: prediction of protein phosphorylation sites with deep learning”. In: *Bioinformatics* 35.16 (2019), pp. 2766–2773.

[20] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective approaches to attention-based neural machine translation”. In: *arXiv preprint arXiv:1508.04025* (2015).

[21] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. “DeepDTA: deep drug–target binding affinity prediction”. In: *Bioinformatics* 34.17 (2018), pp. i821–i829.

[22] Matthew E Peters et al. “Deep contextualized word representations”. In: *arXiv preprint arXiv:1802.05365* (2018).

[23] Alec Radford et al. “Improving language understanding by generative pre-training”. In: *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* (2018).

[24] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI Blog* 1.8 (2019).

[25] Carl Edward Rasmussen. “Gaussian processes in machine learning”. In: *Summer School on Machine Learning*. Springer. 2003, pp. 63–71.

[26] Philip A Romero, Andreas Krause, and Frances H Arnold. “Navigating the protein fitness landscape with Gaussian processes”. In: *Proceedings of the National Academy of Sciences* 110.3 (2013), E193–E201.

[27] Mike Schuster and Kaisuke Nakajima. “Japanese and korean voice search”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 5149–5152.

[28] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural machine translation of rare words with subword units”. In: *arXiv preprint arXiv:1508.07909* (2015).

[29] Jing Tang et al. “Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis”. In: *Journal of Chemical Information and Modeling* 54.3 (2014), pp. 735–743.

[30] Vahe Tshitoyan et al. “Unsupervised word embeddings capture latent knowledge from materials science literature”. In: *Nature* 571.7763 (2019), pp. 95–98.

[31] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[32] Andrew Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269.

[33] Duolin Wang et al. “MusiteDeep: a deep-learning framework for general and kinase-specific phosphorylation site prediction”. In: *Bioinformatics* 33.24 (2017), pp. 3909–3916.

[34] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[35] Ying Xu et al. “PhosContext2vec: a distributed representation of residue-level sequence contexts and its application to general and kinase-specific phosphorylation site prediction”. In: *Scientific reports* 8.1 (2018), pp. 1–14.

[36] Kevin K Yang et al. “Learned protein embeddings for machine learning”. In: *Bioinformatics* 34.15 (2018), pp. 2642–2648.

[37] Zhilin Yang et al. “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: *arXiv preprint arXiv:1906.08237* (2019).

[38] Julian Zaugg et al. “Learning epistatic interactions from sequence-activity data to predict enantioselectivity”. In: *Journal of computer-aided molecular design* 31.12 (2017), pp. 1085–1096.