Predicting protein thermostability change upon single point mutation using deep transfer learning.

Bachelor Thesis

Author: Aaron Máté Spieler (Student ID: 785842)
Supervisor: Prof. Dr. Tobias Scheffer
Co-Supervisor: Dr. Paul Prasse

Machine Learning Group
Institute of Computer Science
Faculty of Science
University of Potsdam

August 28, 2019

Statement of original authorship

I hereby declare on oath that I have made the present work independently and without the use of other than the specified aids. All passages taken literally or by analogy from published and unpublished texts are marked as such. The work has not yet been submitted in the same or similar form or in part in another examination. I certify that the electronic version submitted is in full compliance with the submitted print version.

Aaron Máté Spieler

Berlin, the 26.08.2019

Abstract

Predicting protein thermostability change upon single mutation is a difficult interdisciplinary machine learning task.

Anyone attempting to succeed at it faces many challenges, including finding enough high-quality data to work with, building models that generalize well beyond the proteins that they were trained on, retaining performance while using more widely available features, and especially validating the models in a meaningful way.

While many machine learning architectures like *support vector machines* and *random forests* have been successfully applied to this problem domain, I tried to overcome the challenges and improve model generalization by relying on a state of the art gradient (tree) boosting framework *XGBoost* in combination with a novel transfer learning approach utilizing deep *convolutional neural networks* (*CNN*s), using two datasets, namely *BacDive+* (for pre-learning) and *Merck&Co+* (for dTm prediction). For a more realistic evaluation of performance, I conducted a *leave-one-protein-out cross-validation*, besides the common cross-validation.

While I was able to significantly improve performance over the baseline from *McGuinness et al.* [17] using the features from the *Merck&Co+* dataset, models based on the *transfer-learning approach* alone were not able to compete on the same level. Nonetheless, the latter probably represent the best performing models in literature that do not rely on *3D protein structure* information, and can thus be effectively applied to a roughly three orders of magnitude larger set of proteins. Additionally, the potential for further improvement using transfer learning approaches seems great, and is by no means exhausted.

However, the *leave-one-protein-out cross-validation* also shows, that probably none of these models is suited to *reliably* assist scientists in the wet lab by suggesting thermostability improving single point mutations, as opposed to very recent hybrid knowledge and machine-learning based models published by *Pucci et al.* [24] that seem more promising in that regard.

Zusammenfassung

Die Vorhersage der Änderung der Proteinthermostabilität aufgrund einer einzelnen Mutation ist eine schwierige interdisziplinäre maschinelle Lernaufgabe.

Jeder der versucht sie zu meistern steht vor vielen Herausforderungen, einschließlich der Suche nach qualitativ hochwertigen Daten, der Erstellung von Modellen deren Leistung auch auf vorher ungesehenen Proteinen aufrechterhalten wird, der Beibehaltung der Leistung unter Verwendung allgemeiner verfügbarer Features, und insbesondere, der Validierung der Modelle in einer aussagekräftigen Weise.

Während viele maschinelle Lernarchitekturen wie *Support Vector Machines* und *Random Forests* erfolgreich auf diese Problemdomäne angewendet wurden, habe ich versucht, die Herausforderungen zu bewältigen und die Verallgemeinerungsfähigkeit der Modelle zu verbessern, indem ich mich auf die hochmoderne *Gradient (Tree) Boosting* Architektur *XGBoost* stütze, in Kombination mit einem neuartigen *Transfer-Learning* Ansatz unter Gebrauch von *Deep Convolutional Neural Networks* (*CNN*s) unter Verwendung von zwei Datensätzen, nämlich *BacDive+* (fürs *Vorlernen*) und *Merck&Co+* (für die dTm Vorhersage). Für eine realistischere Bewertung der Modellleistung führte ich neben der üblichen Kreuzvalidierung auch eine *Leave-One-Protein-Out-Kreuzvalidierung* durch.

Während ich durch Verwendung der Features aus dem *Merck&Co+* Datensatz in der Lage war gegenüber der Baseline aus *McGuinness et al.* [17] die Modellleistung signifikant zu verbessern, waren Modelle basierend allein auf dem *Transfer-Learning-Ansatz* nicht in der Lage mitzuhalten. Nichtsdetsotrotz stellen Letztere wahrscheinlich dennoch die leistungsstärksten Modelle die sich nicht auf *3D Protein Struktur* Informationen stützen in der Literatur dar, und können daher effektiv auf einen ungefähr drei Größenordnungen größeren Datensatz von Proteinen angewandt werden. Darüber hinaus erscheint das Verbesserungspotenzial durch *Transfer-Learning* Ansätze groß und ist bisher keineswegs ausgeschöpft.

Die *Leave-One-Protein-Out-Kreuzvalidierung* zeigt jedoch auch, dass wahrscheinlich keines dieser Modelle geeignet ist Wissenschaftler im Labor effektiv durch den Vorschlag einer thermostabilitätverbessernden Einzelpunktmutationen zu unterstützen, im Gegensatz zu kürzlich veröffentlichten hybriden Modellen von *Pucci et al.* [24], die auf Fachwissen und maschinellem Lernen basieren, und in dieser Hinsicht vielversprechender erscheinen.

Contents

1 Introduction

1.1 Objective

The overall objective is, as the title suggests, to predict protein thermostability change upon single point mutation using deep transfer learning.

I will now briefly dissect the meaning of this sentence, since its understanding prerequisites domain knowledge in bio-engineering and specific machine learning (ML) techniques. However, I will focus mainly on explaining the biological terms, since I assume the reader has a computer science background.

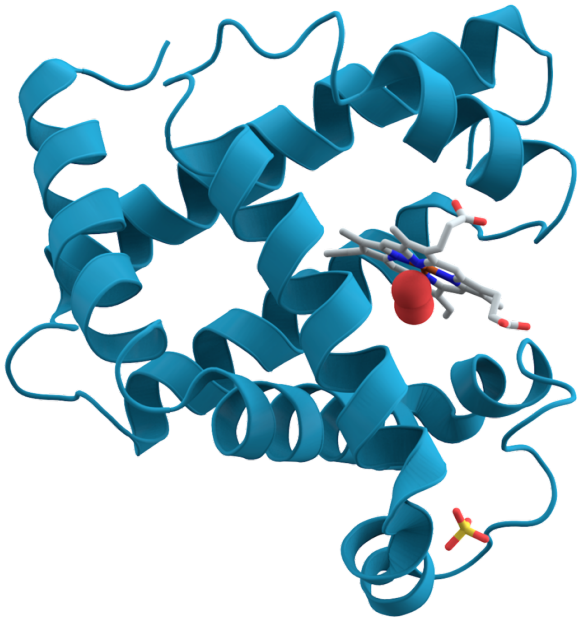

Figure 1: Visualization of the *3D structure* of the enzyme *Myoglobin* [1].

Lets start with what proteins are: "Proteins are large biomolecules, or macromolecules, consisting of one or more long chains of amino acid residues" [31]. In the context of this thesis I only deal with proteins consisting of single amino acid (AA) residue chains, also known as the amino acid sequences, which constitute the *primary structure* of a protein. While the AA sequence will be of prime importance later on, a protein has many more features, most prominently its secondary and tertiary structure (and quartenary structure if its composed of multiple AA residue chains), commonly referred to as its *3D structure*, which as we will see later is the basis for a large portion of the engineered features that are used for this problem setting (see figure 1 for the visualization of *3D structure* of an example protein). Furthermore, two general version of a protein are distinguished in this work; the *wild-type* protein, which refers to the protein in its natural form without any mutations, and the *mutant* or *mutant-type* protein,

which identifies a mutated version of the *wild-type* protein (of which there can be multiple). One prominent example for proteins in biology are enzymes, which control a diverse set of very specific biochemical reactions [31].

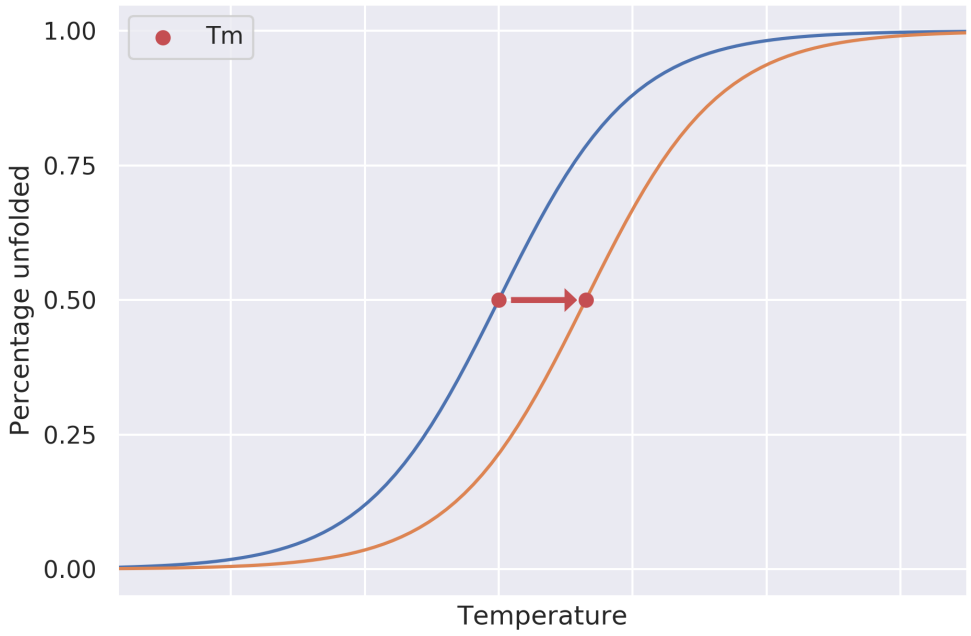

Figure 2: Visualisation of dTm. While the red dots represent the individual melting temperatures (Tm) of the *wild-type* (blue curve) and *mutant* (orange curve) protein, the arrow represents the change (delta) in Tm thus representing a positive dTm in this case.

Thermostability is the next term that requires clarification: While thermostability in general is defined as "the quality of a substance to resist irreversible change in its chemical or physical structure" [32] one way to measure this quality in the case of proteins is the melting temperature (Tm) [17]. The melting temperature of a protein can be defined as "the temperature at which half of the protein population is in a folded state" [17]. Subsequently, thermostability change means the change in melting temperature between the *wild-type* protein and its *mutant* variant, which is referred to as *delta* Tm or dTm short (see figure 2 for a graphical visualization of this definition), which constitutes the target variable (label) of the overall machine learning problem in this thesis.

Single point mutation references to the primary structure of the protein and just means that the difference between the AA sequence of the *wild-type* and the *mutant* protein is a single amino acid. In other words, the *wild-type* protein was mutated (changed) at a single point in its amino acid sequence to another amino acid.

And finally, lets look at the meaning of deep transfer learning, which is a combination of deep learning and transfer learning. While *deep learning* refers to machine learning with artificial neural networks [29] (NN) with multiple hidden layers, *transfer learning* does not have such a clear definition, but generally refers to the problem of learning something in one domain and then transferring the learned knowledge in some way (for example through reuse of weights of NNs) to another related problem domain, and applying it there in hopes of faster learning or better generalization, in short, of better performance [33]. Thus the deep transfer learning means the combination of these two concepts by using deep neural networks for transfer learning. In this thesis specifically, a deep convolutional neural network (deep CNN) is pretrained on the problem of predicting the growth temperature of an organism given a protein from that organism. With extracted knowledge from that network the goal is to predict the dTm for single point mutated *mutant* proteins. This novel approach constitutes a major part of this thesis, which gives this thesis its title.

1.2 Challenges

Understanding the title and thus the objective of the thesis is necessary to understand the overall problem setting, however, it does not explain the decision to pursue the aforementioned deep transfer learning approach.

In order to explain the motivation behind that, and to contextualize other choices made during the project, it is key to understand the challenges of the underlying problem of predicting protein thermostability (upon single point mutation). Thus I will briefly touch upon the key challenges identified during the literature research here.

1. One of the first challenges identified and simultaneously also one of the most severe is the availability of data. There are only a few thousand publicly available data points suitable for dTm prediction at most [17]. This is not much in the field of machine learning, and already an indicator that performance will be severely impacted by this bottleneck. The main reason for this is the incredibly high cost associated with creating data points. Not only does a sequence have to be mutated and then dTm be measured, but also the most relevant and useful features require precise knowledge of the *3D structure* of the *wild-type* protein (which alone is work on the order of one *PHD* or more in biology), which is difficult and time consuming, and thus expensive.

2. Furthermore most available data is low quality. Major databases in this field of research (like *ProTherm*[13]) can be more accurately described as

messy text files containing similar textual information about a collection of proteins. Those databases are not normalized and mainly consist of text field that allowed arbitrary input. The lack of integrity checks and constrains led to many faulty or non-standardised entries. Preprocessing them requires an immense amount of time and domain knowledge, and results in a greatly diminished dataset. [35]

3. Additionally, many previous results and baselines have to be seriously questioned. Either because of the size of the dataset used (on the order of few dozen data points), or the prepossessing used (mostly excluding bad performing data points due to domain knowledge), or because of the faulty or lacking validation (like accuracy reported on imbalanced dataset, or questionable test/train splits, or lack of (even artificial) prospective validation) [17] or simply because the *ML*/prediction methods are insufficiently documented to be able to understand the methodology to reproduce it.

4. Another challenge is to create models that generalize well beyond the *wild-type* proteins the model was trained on [17], which can only really be evaluated using (artificial) prospective evaluation. Simple cross-validation is not enough, since some example *mutants* of of the *wild-type* protein can get into the training set. So far the literature strongly indicates that all models suffer from poor generalization [17].

5. The final challenge is to create models that perform similar to other models in literature, however, use much less (expensive) features so that the models can be used on a larger set of data points in production, for example in protein engineering for a new protein. There are about 153 thousand proteins with some kind of 3D structure information known in the *Protein Data Bank* [2], whereas there are about 141 million proteins with known amino acid sequence in the *Reference Sequence* [20] database of the *National Center for Biotechnology Information*. The best performing models in literature so far use expensive features derived from the known *3D structure* of the protein, however, if similar performance could be achieved with features derived from the *AA* sequence alone, one could apply that model to a thousand times larger set of proteins.

These are the core challenges that need to be addressed by anyone attempting to succeed in this problem domain. Some of them are beyond the scope of this thesis, like the availability of the data (1), which can only be addressed by biologists in the wet lab.

Others, however, like the quality of the dataset (2) can be addressed in at least two ways, either find a high quality dataset or do the prepossessing. While I attempted the latter on the *ProTherm* dataset, which turned out to be futile due to the aforementioned challenges, a more in-depth literature research revealed that there already exists a specifically for *thermostability prediction using machine learning* prepossessed version of it called the *HoTMuSiC* [22] dataset, which was extended by many data points and features by the *McGuinness et al.* [17] paper, whose dataset will be the basis for the work done in this thesis and will be referenced as the *Merck&Co* dataset from here on.

Since the *McGuinness et al.* paper also represents the most recent major publication in this field of research[1], is clean and thorough in its execution and comprehensible in its specif approach, it will serve as my main reference point and baseline, and thus solves the corresponding challenge (3).

The final two challenges, poor generalization of the model (4) and limited usability (5), motivated the deep transfer learning approach. Transfer learning is known to "improve generalization in another setting" [6] which is exactly what is needed in this case (4). And since for the distinct but related domain of predicting organism growth temperature based on a sequence from that organism there is much more data available (on the order of many millions), the choice fell on on a *deep NN* which outperform most other machine learning techniques on large datasets [14]. The specific choice of features used for the *NN*, in this case just the *AA* sequence, also tackles the problem of usability of the model (5). If only based on this network and the extracted knowledge from it good performance was achievable, then it would be usable on a domain about three orders of magnitudes larger than the best performing models in literature (which rely on *3D structural* data) so far.

1.3 Criteria for success

Now that the motivation for this thesis and its context are dealt with and the challenges of this problem domain are known, I will formulate the criteria for success which guided my work:

1. Recreate the results from literature, to create an own baseline and as a starting point.

2. Improve upon those results as measured by common metrics in this problem domain (like mean absolute error *MAE* in cross-validation), once that is done.

[1] During my work on this thesis a newer publication came out, namely *Pucci et al.*[24], which I will reference too.

3. Create models that also generalize better, meaning they also perform good in artificial prospective validation.

4. Create models that are usable on a much larger domain, i.e. try to achieve similar performance with more widely available features.

In the conclusion (see section 8) I will review the results of this thesis in the context of these challenges and will evaluate to what degree they were successfully completed.

2 Related Work

Before I go on to explain in more detail what I have done in this thesis, I would like to give a rough overview of what has been done in literature so far, to give more context to my choices and thus make them more comprehensible.

2.1 Architectures

Even though the literature regarding protein thermostability prediction is quite limited (even more so regarding prediction of Tm or dTm), the range of architectures that have been tried is quite comprehensive.

One prominent approach are knowledge-based architectures that are not (exclusively) machine learning based but rather extensively rely on domain knowledge and the understanding of the intricate correlation between protein features and thermostability or the analysis of statistical correlations between them. [16][22][23][26][28][35][36][24]

Otherwise, more classical machine learning architectures have been tried like *Support Sector Machines* (SVM), *Random Forests* (RF), *Artificial Neural Networks* (which mostly consist of a single hidden layer), *Bayesian Networks* and even *Gaussian Processes* or *K-Nearest-Neighbors* for predicting the the sign of dTm [10][16][17][24][35][36]. The most successful among them are consistently SVMs and even more so RFs.

2.2 Methodologies

Regarding the validation simple train/test splits are common, however, more recent publications consistently rely on cross-validation, in particular *5-fold cross-validation* [22] or *10-fold cross validation* [10], or even *10-trial 5-fold cross-validation* [17]. Unfortunately, only a few papers, including *McGuinness et al.* and *Pucci et al.* [24], did prospective validation, which is a more realistic reflection of performance for protein engineering [17].

Hypertuning on the other hand does not seem to receive any particular attention, only rarely did I find any explicit mention of it or any sections describing methodology on how the hyper parameters were selected, which is an indication of unclean methodology. Nonetheless, some papers do mention (at least a subset of) the hyper-parameters of their machine learning architecture, among others *McGuinness et al.* [17].

Fortunately, the situation for prepossessing on the datasets is much better documented in general, however, it is often based on domain knowledge and/or involves the exclusion of data points from either the train or test set, without

further validation.

Regarding transfer learning I did not find any mention of it in the related literature, which can probably be attributed to the problem of lack of appropriate large datasets (see section 1.2) and the use of only very small artificial neural networks, which made this approach so far infeasible.

2.3 Data

In fact, one of the most prevalent datasets in the domain of Tm and dTm prediction is *ProTherm* [13], thus using any of the derived dataset (like the *HoTMuSiC* dataset or the *Merck&Co* dataset) makes the result comparable to most literature in this field of research. Other very small datasets exist, however, most often the aforementioned dataset is used and supplemented with additional data points. Nevertheless, distinct and larger datasets exist for protein thermostability prediction, but they mostly only contain categorical information like "thermostable" or "mesostable" like the dataset by *Zuo et al.* [37], which cannot be used for the problem setting of this thesis (real valued dTm prediction).

Interestingly, almost all datasets only contain references to the AA sequences, but not the sequences themselves. Thus most datasets only contain features derived from the sequence (like the *amino acid composition*) or features derived from the *3D structure* of the protein. This is probably the reason why there is no model so far that was trained on the amino acid sequence itself as a feature, as far as I know. Additionally, datasets containing point mutations of course contain where the mutation happened in the *wild-type* protein and what the resulting amino acid was at that position, which is usually started counting at 1. Luckily, the *Merck&Co* dataset contains almost all relevant features used in literature so far, except the AA sequence itself.

Regarding the features themselves and their importance a lot of analysis has been conducted already, and thus especially meaningful features for dTm prediction are known, like the number of *hydrogen bonds* and *salt bridges*, the *amino acid composition* (the relative frequency of AAs or AA pairs), the *solvent accessible surface area* or most prominently the ddG (the change in folding free energy between *wild-type* and *mutant*) (whose anti-correlation to dTm is known, but is an expensive feature to simulate/calculate). [17][36][22][10][26]

3 Data

Data is one of the most important factors in machine learning; its retrieval and prepossessing constitute a major portion of the time invested in the problem setting and its quality and quantity fundamentally limit what can be achieved.

In previous sections, especially section 1.2 and 2.3, I thoroughly dealt with contextualizing the situation of data in the given problem setting of this thesis, thus the focus of this section will be the specifics of the datasets used in this work and how they were prepossessed.

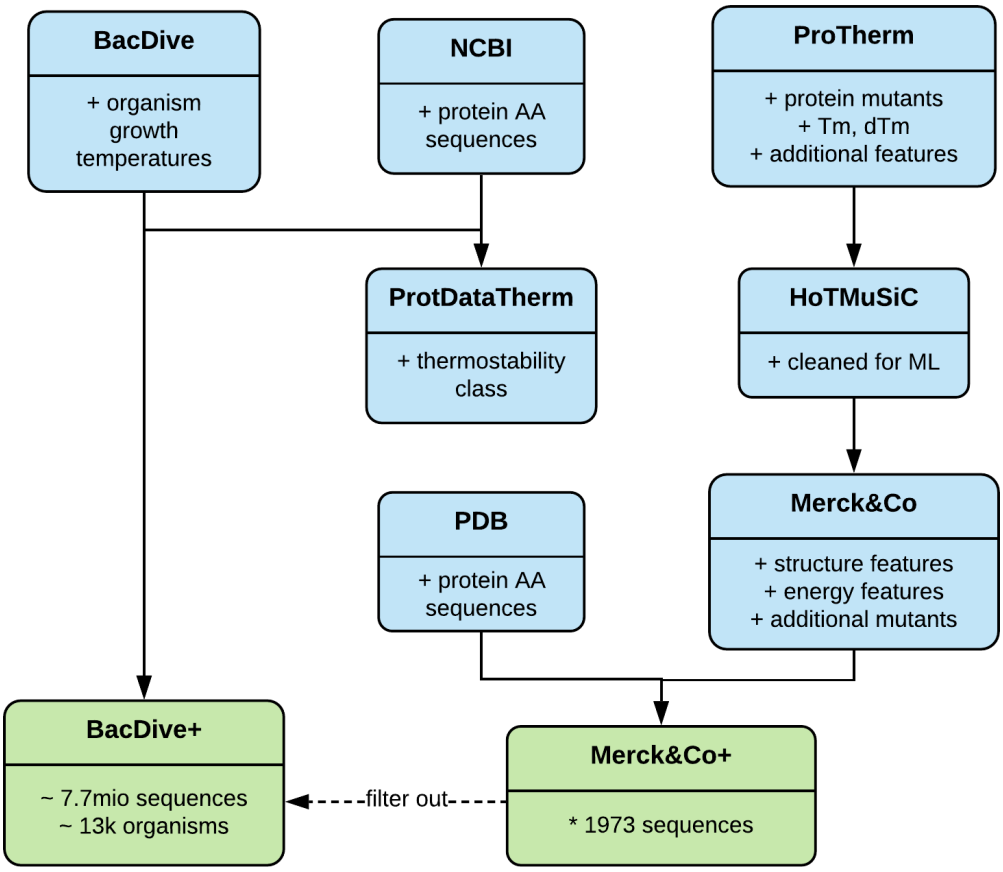

Figure 3: Overview of the datasets used in this thesis, their core features in this context, and their relation to other datasets known in this area of research. Datasets that I preproccessed and used for models whose performance is reported in this thesis are marked green.

3.1 BacDive+

This dataset was *only* used for pre-learning models that could then later on be exploited for transfer learning to tackle the main problem setting that is predicting change in protein thermostability upon single point mutation.

BacDive+ contains pairs of AA sequence and real valued information about the *averaged organism growth temperature range* ($AOGTR$) of the organism that the protein was found in. The "+" is supposed to signal the addition of the AA sequence information to the information found in *BacDive*.

Its unpreprocessed version was created[2] from two other datasets; *BacDive* [25] (which contains information about bacteria and their properties, like their *((optimal) growth temperature ranges*) and *NCBI* [20] (which contains information about proteins, for example their AA *sequence* or the organism they were found in). Its creation was strongly inspired by the dataset called *ProtDataTherm* [19], however, differs from in in a significant way, in that it contains the real valued $AOGTR$ and not the thermostability category (like psychrophilic, mesophilic or thermophilic) of the organism the protein sequence is from. This enables the use of this dataset for a regression problem and gives more options for preprocessing.

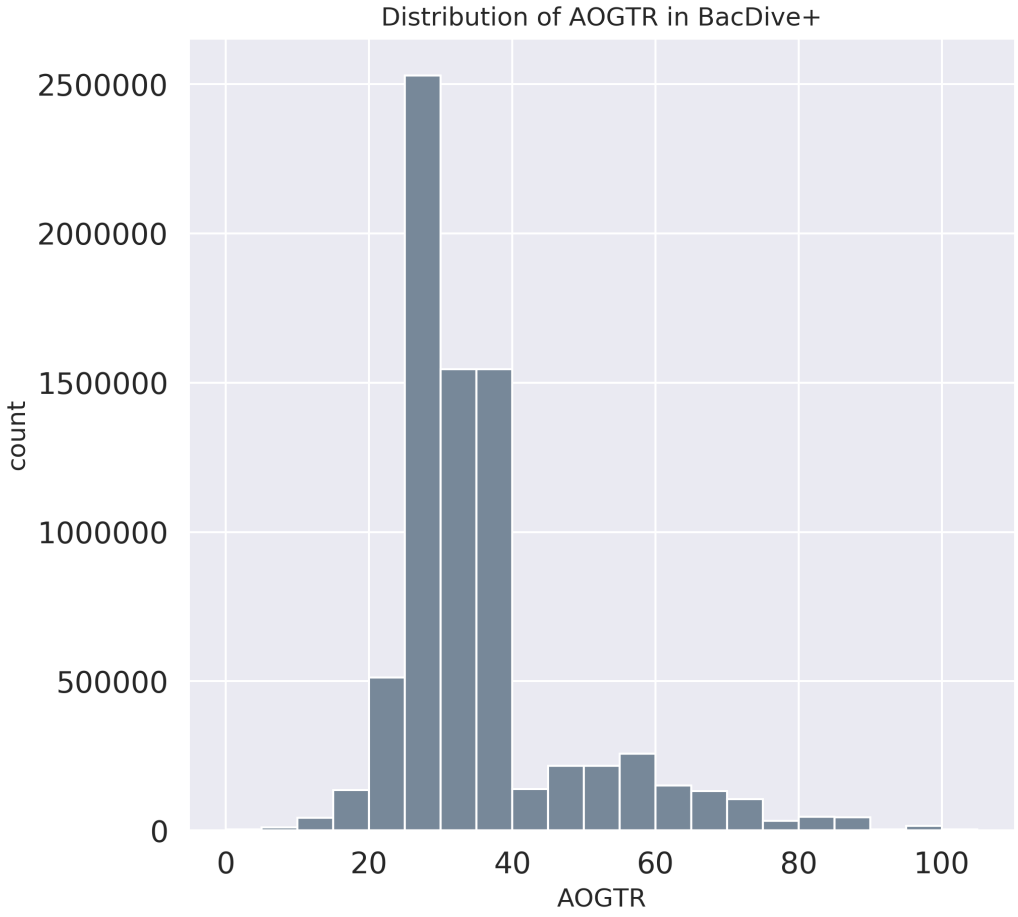

Figure 4: The distribution of the *averaged organism growth temperature range* ($AOGTR$) in $°C$, with an average of $31.15°C$ and a standard deviation of $7.76°C$, on a per protein basis.

[2]By Lukas Golombek of the *iGEM Potsdam 2019* team

The preprocessed version of the dataset that was used for *AOGTR* prediction contains 7.701.283 sequences from 28.960 (mainly mesophilic) organisms. The distribution of the *AOGTR* label can be seen in figure 4.

3.2 Merck&Co+

This dataset is used directly to tackle the core problem setting of dTm prediction upon single point mutation and is mainly derived from the *Merck&Co* dataset, which most importantly contains information about the *wild-type* protein, its mutated version and the resulting dTm. *Merck&Co+* only really differs in that it additionally contains the *AA* sequences themselves, rather than only an *ID* referencing them, and the mutated sequences, rather than only the position and type of the mutation. These features were a necessary addition, since the transfer learning approach described in section 4 utilizes the *AA* sequences themselves directly using a deep *CNN*.

Merck&Co+ consists of three subsets of data: *T1626* (also "training" or cross-validation dataset; derived from *HoTMuSiC* [22]), *T251* (also retrospective dataset) and *T96* (also prospective dataset), where the numbers also represent the number of data points in them [17]. Together they form *T1973* (also leave-one-protein-out cross-validation dataset). For an in depth explanation of the origin of these datasets and their features please refer to the *McGuinness et al.* h[17] paper.

Nevertheless, I will now briefly touch on the features found in the *Merck&Co+* dataset as in so far they are important to understand the types of features contained in the dataset which constitute a significant part of the taxonomy of the models (which is explained in detail in section 5.1). Abbreviations relevant for future reference are marked bold.

The categories are the following:

1. **S** (structure): these are all the features that are related to the sequence structure of the protein in question. It can be divided into three subcategories:

   (a) **A** (amino acid): these features are concerned with the amino acid where the mutation happens ("_first") and the resulting amino acid at that position ("_second") and their change in properties ("_diff"). Examples features include *charge*, *mutability* and *polarity*. [17]

   (b) **L** (local): these descriptors were calculated for the wild-type protein and include secondary structure information but are mostly about fea-

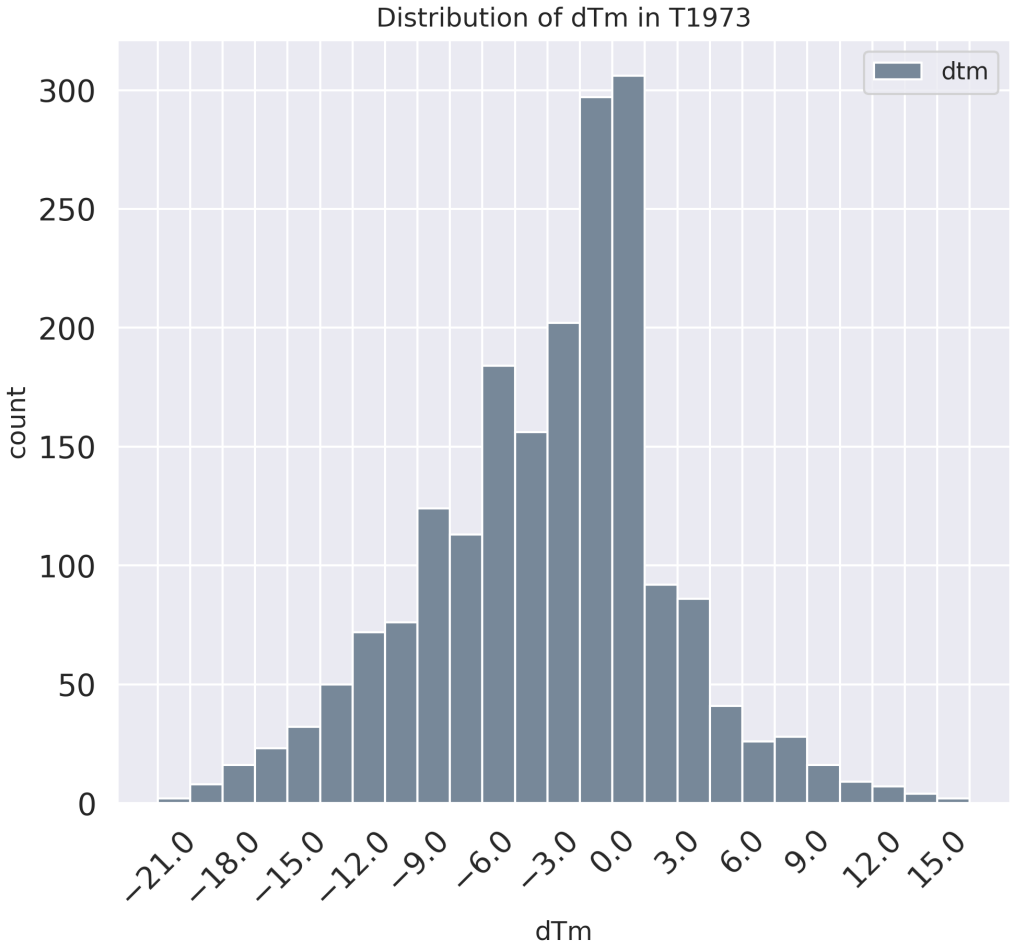

Figure 5: The distribution of change in melting temperature (dTm) in $°C$ between wild-type and mutant sequence in the *T1973* dataset. The plot for the *T1626* looks very similar. While the *T251* datasets dTm distribution looks comparable (with noticeable gaps next to the mean), the *T96* dataset has a strong bias around $-9°C$. See the appendix A for the individual plots.

tures related to the *solvent accessibility* of the protein like *fraction buried* or *hydrophobic area*. [17]

(c) **G** (global): features describing the whole protein and its overall *3D structure* are situated here and include the likes of *mobility*, *net charge* and *dipole moment*. [17]

2. **E** (energy): these features are all *ddG* estimates or features used to derive *ddG*. They are calculated using 5 different libraries: **BL**, **DS**, **CART**, **MOE** and **MONO**. [17]

3. **EXT** (extra): a subset of the features that are only in *T1626* because they were carried over from *HotMuSiC*. Examples would be the Tm of the wild-type sequence and the pH at which measurements were made.

Other than that there are several data files that are aligned with the *Merck&Co+* dataset and contain extracted features from the deep *CNN* utilized for transfer learning (due to the specific methodology employed) that are not part of the *Merck&Co+* dataset per se, but could be used for further research.

3.3 Preprocessing

Since both datasets contain AA sequences, some preprocessing steps were the same and necessary for compatibility and applicability of the transfer learning method. I will start by describing those and move on to the steps specific to each dataset.

General preprocessing on both datasets:

- Sequences shorter than 50 or longer than 650 residues have been removed from both datasets. This corresponds to about one standard deviation around the mean sequence length in the original *BacDive+* dataset (resulted in the removal of around $700k$ sequences), and applies to each AA sequence in the original *Merck&Co+* dataset.

- Sequences that contained *non canonical amino acids* were removed as well. This affected a few thousand sequences in *BacDive+* and none in *Merck&Co+*. This has been done since there are no examples of such sequences in the *Merck&Co+* dataset, and because of the rarity of such sequences in *BacDive+*.

- All sequences have been encoded as integer arrays of length 650 where 1 through 20 represent the canonical AAs and 0 represents "None". In the usual case that a sequence was not of appropriate length it has been filled up with trailing 0s.

Preprocessing specific to *BacDive+*:

- Different temperature aggregates have been calculated for the dataset (which mainly contained the *organism growth temperature ranges*) like the above mentioned *averaged organism growth temperature range* ($AOGTR$[3]) or the *maximum organism growth temperature* ($MOGT$).[4]

[3]Due to the lack of growth temperature range information the $AOGTR$ could not be calculated in a few dozen cases; corresponding sequences were excluded for $AOGTR$ prediction.

[4]The organism growth temperature was usually given by a range of (min,max), however, sometimes also a second range was given or even a single value. The AOGTR was constructed by flattening the array of given ranges and values and then taking the arithmetic mean. This gives more weight to the specified ranges if a range and a single value was specified. For the maximum of the growth temperature range simply the single highest value was taken.

- Finally, sequences that were also contained in the *Merck&Co+* dataset (exact matches) were removed from *BacDive+*, which resulted in the removal of 88 sequences.

Due to the architecture of the deep *CNN* deployed (see 4.1) no further preprocessing was necessary (like one-hot encoding the *AA*s).

Preprocessing specific to *Merck&Co+*:

- Most importantly, the actual *AA* sequences for the *wild-type* proteins have been retrieved using the *PDB-ID* and mutated at the given location to the specified amino acid to create the *mutants*. In the case of some proteins[5] (all from the *T1626* dataset) specific pre-mutations to the *wild-type* protein were specified [22] and thus applied. For some *mutants* the specified mutation did not match at the given location, however, this was rectified in most cases by using the offsets stated in the *PDB* [2] database [6]. The remaining mutations were also rectified, however, due to the not so straight forward nature of the rectifications[7] they were sometimes excluded from the dataset (marked by **_f** in the graphs) to check the impact of potential errors on performance. In all cases the final offset that was used to apply the mutation is included in the *Merck&Co+* dataset. There were no complications in applying the mutations in the *T251* or *T96* datasets.

- An important step was the assignment of features in the dataset to the feature categories specified in the paper or by me (only *EXT*) as stated in 3.2. The feature names were then made consistent (capitalization and ordering) across the different data subsets. Some features were excluded for various reasons [8]. Some global features (*G*) that should be in the dataset (according

[5] specifically *1BNI_H102A*, *1URP_L265C*, *1YCC_C102A*, *5PTI_M52L*

[6] specifically *1H7M*, *1SHF*, *1YCC*, *1FNA*, *1GV5*, *1I4N*, *1JNX*, *3UUE*, *3SIL*, *1MJC*, *1KE4*, *1KFW*, *1YU5*, *4BLM*, *1YCC*, *1FNA* (Additionally, *1BNI* and *1AYF* turned out to be in this category upon later review)

[7] Most of them (*3KS3* (2), *1TPK* (21), *4BLM* (4), *1YEA* (1)) were rectified by partially applying the mutation at the offset specified by the *PDB* database -1. I presume these peculiar offsets might come from the inconsistent indexing. In the case of *1TPK* the index 0 does exist, in the case of for example *1H7M* is does not, and indexing jumps from -1 to 1 directly (as it does in most cases in *PDB*), even though both sequences have negative offsets. Upon review, it turned out that for *1YEA* (1) the specified *PDB* offset was correct and applying an additional -1 was incorrect. Additionally, it turned out that I incorrectly put *1BNI* (34) and *1AYF* (18) in this category, even though all mutations were correctly applied using the specified *PDB* offset. The numbers in the brackets specify the number of mutants affected out of a total of 80 mutants in this category.

[8] *wtss* was excluded (for the prospective validations) because its not available in *T251*, *variantpolarity* because its a redundant categorical feature. From the features that are there for legacy reasons from *HoTMuSiC* only a subset that was deemed usable was utilized in *EXT*, which is only available in *T1626* and was only used for the cross-validation performed on *T1626*.

to the description provided with the dataset) seem to be missing, like *dipole direction* (maybe because the questionable usability for machine learning?) or *protein mass*. The same seems to be the case for some features of the *MONO* category. Otherwise, the assignment was complete and successful.

Due to the nature of the machine learning architectures deployed for this problem (see 5) no further preprocessing was needed on *Merck&Co* (like feature normalization or feature selection).

4 Transfer Learning

Since the general idea and benefits of transfer learning have been discussed in section 1.1 and 1.2, I only want to elaborate on the core relation of the transfer learning problem to the main objective here, before I move on to the detailed description of the transfer learning approach in this thesis.

The specific problem setting for the pre-learning is the following: "given a primary *AA* sequence, predict the *AOGTR* of the organism that the protein was found in". The relation to the objective of this thesis is the following:

1. A high *averaged organism growth temperature range* (*AOGTR*) implies that the organism grows at higher temperatures.

2. For an organism to grow at higher temperatures, its proteins need to work at higher temperatures.

3. If a protein works at a high temperatures, it necessitates that its not melted [31], implying an even greater melting temperature (*Tm*)[9].

4. Thus (following from 1 through 3) a high *AOGTR* generally implies a high *Tm* for the proteins in organism (the implication is intended to be read as describing a statistical correlation, and not a rule without exceptions).

So a deep *NN* that learned the correlation between certain protein properties, like specific *AA* sequence patterns in the protein, and the *AOGTR* associated with it, should also have indirectly learned something about the correlation between those features and the proteins *Tm*.

This indirect knowledge can then be extracted and used to predict the *Tm* for a single protein, or the *dTm* for *wild-type* and *mutant* protein pairs, which is the main objective of this thesis.

A final note: the specific objective of the pre-learning was completely determined and directly implied by the available data on proteins (which was discussed in section 1.2). This includes the availability of protein properties that could be used as features, and the availability of temperature information on those proteins that is somehow correlated to the *Tm* of the proteins, with key focus on the amount of data available. While *ProtDataTherm* essentially has the necessary characteristics, the most suitable dataset was *BacDive+*, since it is higher quality and contains real-valued label.

[9] See section 1.1 for the definition of melting temperature of a protein

4.1 Architecture

As stated in section 1.2, artificial neural networks are a common choice for transfer learning problems. So due to their strong performance in many machine learning areas, for example image recognition, natural language processing or sequential data analysis [14], and due to their straight forward usability for transfer learning they were perfectly suited for this pre-learning problem.

The specific choice for the overall network architecture fell on a convolutional neural network (CNN)[10] for 3 main reasons:

1. First of, through *Google Colab* [7] I had accesses to significant free computational power in the form of *cloud TPUs* that can be utilized particularly efficiently by CNNs [11].

2. CNNs are easily used for transfer learning; either by fixing layers and continuing training, or by extracting representations of the input at different layers, which can then be used in another architecture.

3. Most importantly, however, the only suitable feature available at a scale appropriate for deep learning was the primary AA sequence of proteins.

   (a) And CNNs are a very popular choice for a deep learning architectures for discriminatory problem settings where spacial correlations are of the utmost importance, most prominently in image recognition where they are state of the art and achieve top performance [12][14]. In this case the 1D CNN would hopefully learn to recognise specific AA sequence patterns

   (b) While other features could have been engineered in addition to it, like the features on a per amino acid basis, a deep CNN should be able to implicitly learn them with enough data.

Thus availability, and especially the type of the available feature, was the central determining factor for the network architecture, which is one of the core challenges as discussed in section 1.2.

The last preemptive architecture choice was regarding the input layer or rather the first layer after that. The choice fell on an *Embedding Layer*. This type of layer allows for the network to learn its own representation of specified length for each category, which is more powerful and efficient than a simple one-hot encoding layer.

[10] Alternative architectures like fully connected NN have also been tested by other members of the *iGEM Potsdam 2019* team, and performed significantly worse in pretesting.

Other than that, a base architecture (see appendix A fig 20) was used and later hyper-tuned. The base architecture was derived from best practices in literature [8][15][21][12] and from previous experience on a similar problem setting that involved the *ProtDataTherm* dataset (which is not discussed in this thesis).

4.2 Methodology

This section is mainly about the training, tuning and validation methodology of the pre-learning.

Due to the relatively large dataset (about 7.7 million data points) a simple *Triple Cross Validation* was performed with one fifth of the data belonging to the holdout set, and two thirds of the rest belonging to the train set, and the remaining to the tune set (see figure 6).

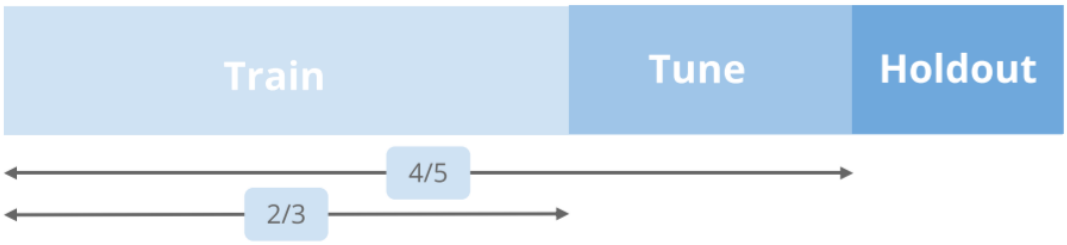

Figure 6: Visualization of the relative proportions of the train, tune and holdout set sizes for the methodology employed throughout the thesis.

Before the split, the whole dataset was deterministically shuffled. The tune set was used for hyper-tuning[11], and the holdout set for evaluating the performance of the models.

For the reported holdout-set performance (see section 4.3: **_h**) the model was trained separately on the 3 possible inner train/tune splits and each time evaluated on the holdout set (3-fold). Similarly, the test-set performance was calculated by training on the three possible inner splits, however, then evaluating on the tune set (see section 4.3: **_t**).

Additionally, the target labels were *z-score normalized* using the appropriate train set each time. Scaling with the train and tune set has also been attempted (see section 4.3: **best_scale+_t**). To check whether performance improves with more data, the final architecture was trained on the train and tune set (see section 4.3: **train+**).

The default optimization criterion was the *Mean Squared Error (MSE)*, however, the *Root Mean Squared Error (RMSE)* and *Mean Absolute Error (MAE)*

[11]It was also used for early-stopping, for which a separate early stopping set could have been used, but due to the large sample size it did not seem necessary. Also it turned out that most models stopped after around 12 epochs.

are reported and compared in section 4.3.

4.2.1 Hypertuning

Hypertuning is a tedious and very resource intensive task, especially in the case of deep learning. Even with the access to a *cloud TPU*, a nested cross-validation would not have been feasible, additionally, due to the pre-learning task itself not being the main objective of this thesis, I opted for a simpler and more environmentally friendly *systematic manual tuning*.

In tuning method was comprised of 5 steps:

1. Use baseline architecture as stated in section 4.1
2. Create new model architectures by changing single hyperparameter of baseline architecture
3. Create model architecture by combining significantly better performing hyperparameter changes from previous step
4. Create new model architectures by changing single hyperparameter of model architecture from previous step
5. Select best performing model architecture

So the hypertuning roughly consisted of two rounds of improvement with a mixing round in-between, which can be regarded as overall quite limiting.

To assess the significance of improvement, the calculated risks were compared to the 95 percentile confidence interval of the model from the previous hypertuning step. The risk itself was not evaluated in a three fold fashion as the models whose performance is reported in section 4.3, but on the same train/tune split, which I deemed representative enough due to the low variance measured in the performance on the different splits in the case of the baseline model.

The hypertuning-parameters included, but were not limited to: label *z-score* normalized vs not, optimization criterion *MSE* vs *MAE*, batch size 1024 vs 512, embedding layer size 8 vs 4, first layer *kernel size* 7 *and stride size* 2 vs *kernel size* 3, pooling layers *max* vs *average*, global pooling layers *max* vs *average*, 3 vs 4 *CNN* layers per convolutional block, 2 vs 3 convolutional blocks, doubling the *filter counts per convolutional block* vs not, no feed forward concatenation vs *ResNet* type feed forward concatenation, activation *ReLU* vs *ELU*, no drop out layer vs drop out layer. The former choice represents the default network parameter, whereas the latter the alternative that was tested.

4.2.2 Knowledge Transfer

To better understand the choice of architecture for the main problem setting of this thesis (see section 5), I will now briefly discuss the approach to transfer learning that I pursued.

In the case of this deep *CNN* the most straight forward way to do transfer learning would be to just add another layer before the output layer to the network, fix the other weights, and retrain the network on the new domain. However, due to the availability of other features through the *Merck&Co+* dataset, whose incorporation into such an architecture would have complicated things significantly, but most importantly, due to preliminary testing of this approach, which indicated bad performance (results not displayed here), I rather opted for the approach of extracting knowledge from the deep *CNN* in the form of float arrays, which were then used as input features to the models described in section 5).

Specifically, each *wild-type* and *mutant* sequence in the *T1973* dataset was used as input to the *best* deep *CNN* from section 4 (see figure 7), and its representation in specific network layers was extracted (as indicated by figure 7) and then saved to a different, but aligned *csv* file[12]. The contents of these files were later used as features for the dTm prediction. For the exact details of which extraction layer was used and for which sequence-type, refer to the taxonomy section 5.1.

This method of transfer learning is easily extensible with features from other datasets, since knowledge is transferred in the form of new features (the extracted float arrays), that could be used by any other model architecture. This way of transferring knowledge/transfer learning allows for a different choice of architecture for the main problem domain, and in this particular case made it possible to use gradient tree boosting (see section 5), which is much closer to *random forests*, which have been proven to be successful in thermostability prediction (see section 2.1).

4.3 Results

In this section I will briefly review the best performing deep *CNN* architecture that resulted from the hypertuning (see figure 7), and discuss the results from the pre-learning based on figure 8.

Most noticeably the *CNN* has double the filters per *convolutional block* and

[12]The extractions are all uncorrelated, since the deep *CNN* network and its weights were fixed for all extractions. The extraction was done by throwing out all layers after the extraction layer, which lead to the extraction layer being the output layer. The output that was generated when a specific sequence was used as an input, and one forward propagation was performed, was then simply saved to a *csv* file.

a *ResNet* type feed forward concatenation from the middle of the network to the dense layers, of which there are now two with twice the number of neurons per layer. Additionally, the *GlobalAveragePoolin1D* has been switched out for *GlobalMaxPooling1D*. Additionally, the *batch size* has been halved. Other than that, the network is the same as the base network.

The performance of all relevant models is depicted in figure 8, most importantly that of the absolute baseline (*avg_pred*), which is predicting the average T calculated on the train and tune set for the holdout set, the default architecture (*default*) and the final architecture derived by hypertuning (*best*).

As can be deduced from figure 8, even the *default* deep *CNN* performs vastly better then predicting the average (about 8.91 *MAE* vs 6.24 *MAE* (*_h*)). With the additional hypertuning, the performance was improved significantly (which is implied by the relatively small standard deviations) by around .53 *MAE* to about 5.71 *MAE*. Interestingly, test-set performance was almost identical to holdout performance, which can probably be attributed to the large sample size of both.

The good performance of the default model and the comparatively small improvement through the hypertuning indicate that the the architecture based on previous experience and best practices was already a solid design. The particularly bad performing average prediction could also be attributed to the bi-modal distribution of the target labels, maybe predicting the average of the dominating peak would have yielded a better baseline. In any case, more sophisticated hypertuning methods and more computational resources still seem promising in regards to improving performance further.

The *best_scale+_t* model shared the same architecture with the *best* model and was used to test whether using additional data (by also utilizing the tune set) for scaling would lead to improved performance. As can be read from the graph it did not, which was expected due to the already large train dataset.

To get a rough estimate of the probable performance of the models used for transfer learning, which were trained on train plus tune set, their performance was evaluated on the holdout set, which they also utilized for early stopping. Since the train and tune set performance seems to be almost identical in previous measurements, I expect this estimate to be pretty accurate. As we can see with this model performance can potentially be improved significantly by utilizing more data. For future experiments one could try to use all data for training with a constant 12-15 epochs of training, which seems to be the range of epochs in which all models stopped training, instead of using early stopping.

4.4 Reflection

Overall, this pre-learning problem of predicting the *AOGTR* of an organism given the primary *AA* sequence of a protein from that organism seem to be a viable machine learning problem on its own, and the choice of a deep *CNN* architecture seems to be justified by the good performance it achieves.

Even though better performance can probably be achieved as discussed in section 4.3, the *systematic manual tuning* seems to have worked well too. It might be worth considering other architectures as well, or extending the *BacDive+* dataset with more examples. Additionally, introducing class-weights for training, or data sub-sampling might be useful considerations due to the imbalanced dataset.

While the success of this pre-learning problem stands on its own, its usefulness in the context of this thesis can only be judged with regards to the overall problem setting, which will be evaluated in section 7, and remains to be seen.

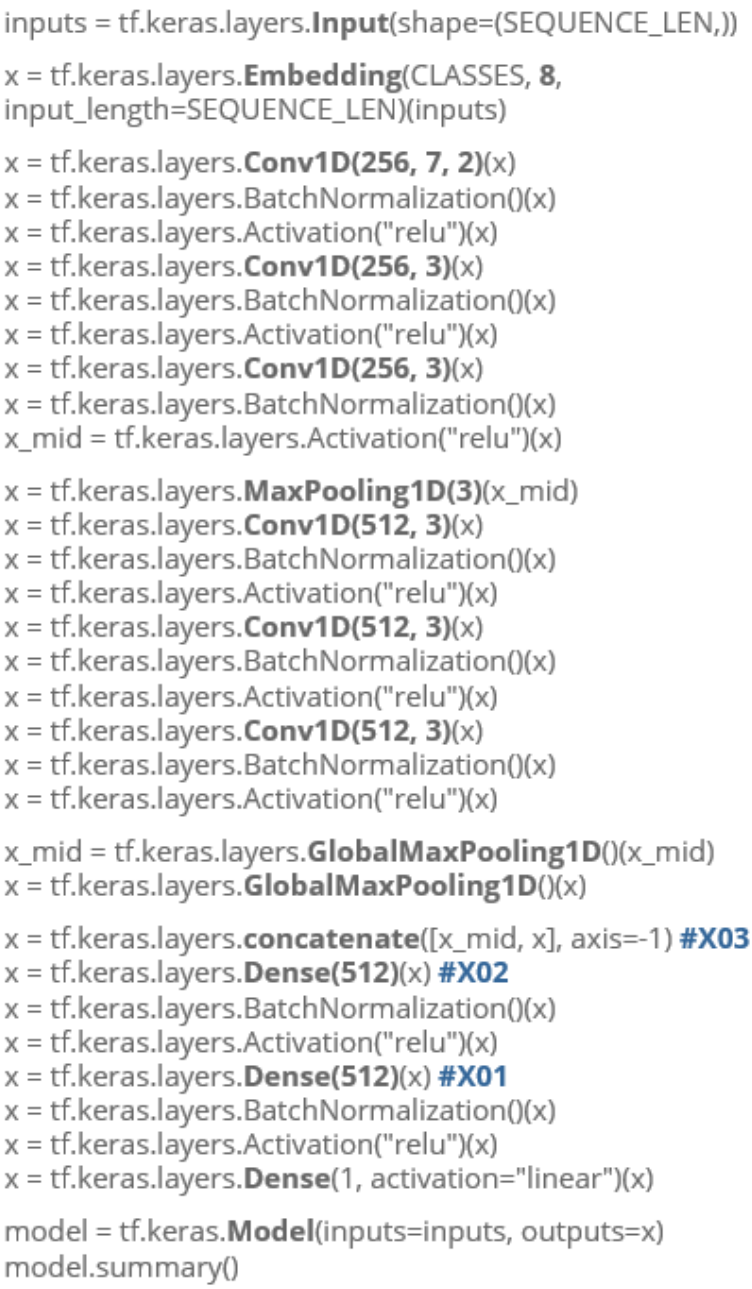

Figure 7: The final deep *CNN* architecture. The relevant extraction layers for the transfer-learning have been marked in blue with #**X01**, #**X02** and #**X03** for future reference in 5.1. It was trained with *MSE* as the optimization criterion, with target labels *z-score* normalized, and a *batch size* of 512.

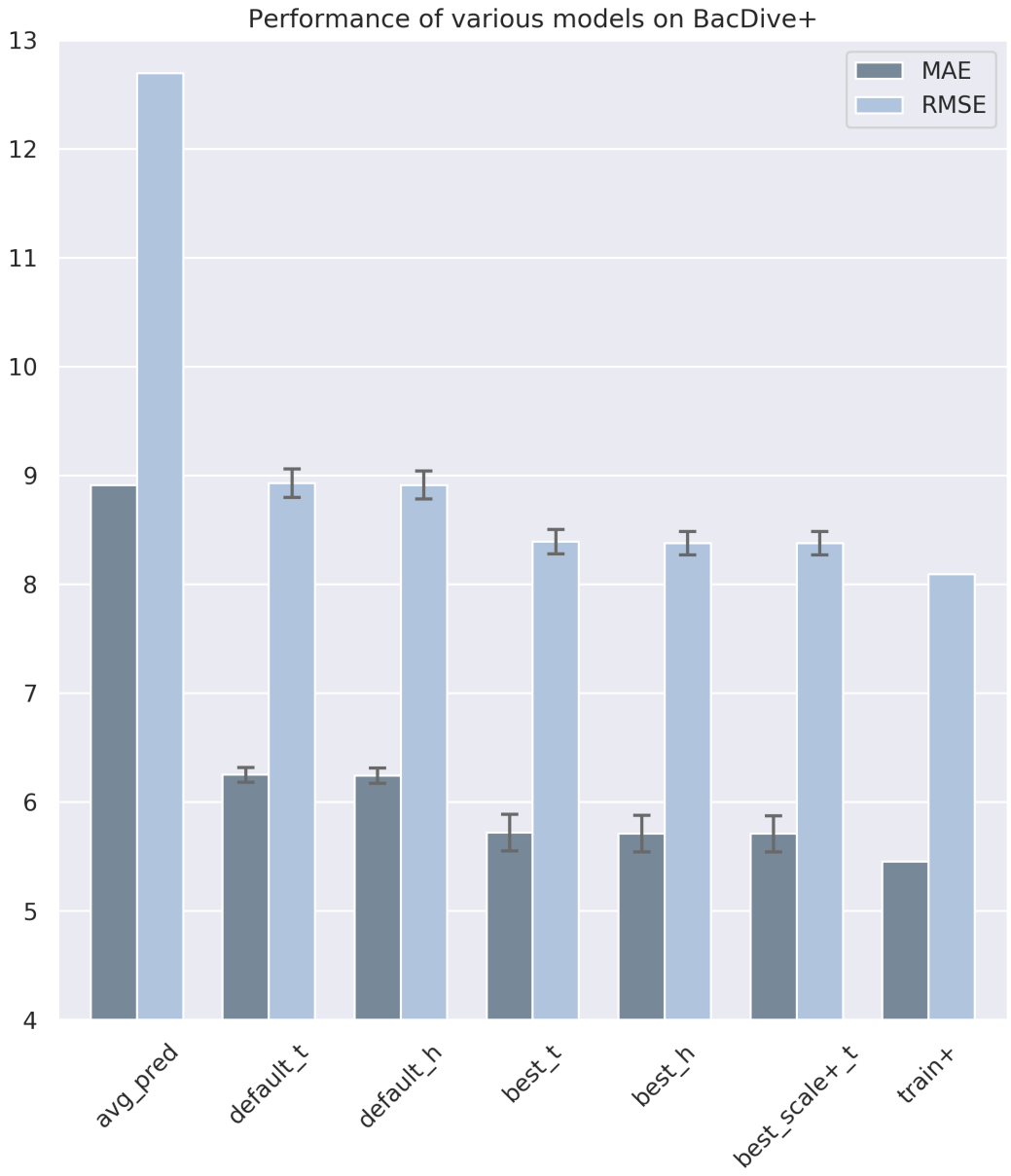

Figure 8: This bar-chart displays the performance of various models on the *BacDive+* dataset. The error bars represent one standard deviation where the measurement was done in a 3-fold fashion. While **_t** represents the test set performance, **_h** represents the holdout set performance. In the case of **best_scale+_t** the same model as **best** was used, however, the label was normalized for training with the use of the train and tune set. Using the train and tune set as a training set has also been attempted with the **train+** model. For the complete table of results see table 1 in the appendix.

5 Architecture

This section deals with the description and motivation of the core architecture that was used, and with the model taxonomy that was applied to all the models that were created to tackle the main objective of this thesis, specifically, for the problem setting of predicting change in protein thermostability (in the form of dTm) upon single point mutation.

The model architecture deployed, or rather machine technique used for this problem setting is called *gradient tree boosting* (using the *XGBoost* library). The choice is mainly motivated by the related work, the specific advantages of this method over the previous techniques, and the choice of transfer learning methodology, more concretely:

1. *Random forests* were one of the two consistently best performing architectures in the thermostability prediction literature (see section 2), which is very closely related to *gradient tree boosting* in the sense that both are ensemble learning methods based on decision trees (or stumps). [30]

2. Additionally, gradient (tree) boosting performs much better in most use-cases, because it leverages information about previously made errors (through residuals), wheres the weak learners in random forests are uncorrelated. This gives it the additional advantage of bias reduction, besides the variance reduction due to bagging, and other advantages like insensitivity to useless or collinear features due to being decision tree based. [30][5]

3. The gradient tree boosting method requires very little preprocessing compared to alternative machine learning architectures like SVMs, due to it being a decision tree learning method, which generally require little preprocessing. [30]

4. And finally, the structure (S) and energy (E) features provided through *Merck&Co+* can be easily extended with the extracted representations from the deep *CNN* of the different AA sequence types to be used with a gradient tree boosting architecture.

Beyond that, the concrete choice of decision tree boosting method fell on the state of the art *XGBoost* library for its performance and better regularization compared to other popular libraries like *AdaBoost*, and due to it achieving top performance in current machine learning competitions. [3].

The models were constructed using the tree booster and consist of 1000 trees. The specific choice of hyper-parameters beyond the default parameters are de-

scribed in the next section under hypertuning because they were automatically selected for each cross-validation fold.

5.1 Model Taxonomy

Since there were many models tested with different hyperparameter, which used many different subsets of features and data, this subsection is dedicated to the description of the model taxonomy.

Most model names can be described with a single regular expression:

$$r"(S)?((\hat{}|\_)(E|BL|DS|CART|MOE|MONO))?(\_EXT)? \\ ((\hat{}|\_)(W|M|D|A)(1|2|3)(1|2|3)?)?(\_s|\_f)?" \quad (1)$$

The majority of abbreviations should be familiar from section 3.2. The model can use features from the *Merck&Co+* dataset like structure features (S), all energy features (E) or energy features from specific libraries only (BL trough $MONO$) or even use the extended features (EXT).

The model can additionally, or exclusively use features extracted from the deep *CNN* using the wild-type sequence (W) or mutant sequence (M). It is also possible to use the calculated difference of the two extractions (D). If W, M and D are all used together it is marked as A. Additionally, the extraction layers have to be specified, which can be a range of layers or just a single layer.

The labels can be scaled with *z-score* normalized during training (s), and finally, it is also possible to only utilize the reduced (filtered) dataset (f) (see section 3.3).

Additionally, for some testing, instead of the representations that were extracted from the deep *CNN*, the unprocessed raw sequences were used for a comparison, which is denoted by **raw_** (see figure 5). Beyond this, a special baseline model is the average prediction **AVG**.

One last remark; the models that were cross-validated on the *T1626* dataset fall into three general categories: models that only use features from the *Merck&Co+* dataset (see figure 9), models that only use the extracted deep *CNN* features (see figure 10), and models that use both (see figure 11).

6 Methodology

Cross-validation on the *T1626* dataset:

As discussed in section 1.2 and 2.3 one of the most meaningful evaluations

was measured on the *T1626* dataset by a *10-trial 5-fold cross-validation* (see the *McGuinness et al.* paper [17]) and thus served as my baseline.

For the purpose of cross-validation the *T1626* dataset was split into a train, tune and holdout set, with the same proportions as during the pre-learning (see section 4.2, figure 6), however, instead of a *triple cross-validation* I opted for a *7-trial 5-fold cross-validation*[13] due to the comparatively very small dataset. The hyper-parameters were automatically selected in each round with a *3-fold bayesian optimization*.

The 7-*trial* 5-*fold cross-validation* comes down to deterministically shuffling the dataset seven times with different seeds and then performing a 5-fold cross-validation each time. The arithmetic mean of the *MAE*s and *RMSE*s of these 35 runs is then reported along with its standard deviation as the performance of each model.

Regarding the feature importance: To evaluate which specific features were the most important for the prediction and how they correlated with the dTm, the feature importance was measured using the *SHAP* library[14] and displayed for the 20 most important features for selected models, which were trained on the whole *T1626* dataset for this purpose. This allows for the comparison with current literature and further investigation regarding the extracted deep *CNN* features.

Regarding the sanity check: The sanity checks were also performed on the *T1626* dataset in the cross-validation fashion. The goal was to evaluate whether the representations extracted from the deep *CNN* actually provided any value over just using the raw sequence for example, which should be the case if the transfer learning worked. Additionally, the contribution of the specific type of extracted representation (which layer, and whether wild-type or mutant or both) was investigated.

Leave-one-protein-out cross-validation on the *T1973* dataset:

As mentioned before, the *McGuinness et al.* paper is one of the few papers that performed prospective validation, for which they used one of their models trained on the *T1626* dataset to predict promising *mutants* based on a new *wild-type* protein, which were then created in the wet-lab, which resulted in the creation of the *T96* dataset.

Unfortunately, with the available resources a prospective validation was not

[13] I later realized that the *McGuinness et al.* paper went for a *10-trial* and not a 7-*trial* cross-validation, which ultimately lead to very similar standard deviation as can be seen from the results in figure 3 and 4 of the paper [17]

[14] The *GitHub* library at *slundberg/shap*[27] was used for this purpose.

possible at the time. Nevertheless, in the hopes of more meaningful results, the *leave-one-protein-out cross-validation* was performed. For that the *T1973* dataset (with a total of 134 different *wild-type* sequences) was also split in three parts, however, instead of each part taking a certain percentage, the holdout set always consisted of all the *mutant* variants of a single *wild-type* protein. The rest of the data-points was then used as the training set, on which a *3-fold bayesian optimization* was performed for the hyper-parameter selection.

This kind of validation was supposed to be a good proxy for a true prospective validation, since in both cases there are no examples available to the model about *mutants* of the target *wild-type* protein. For this reason it could also be referred to as an *artificial prospective validation*. Interestingly, this kind of cross-validation was not performed in literature so far, even though it should give a much more accurate picture of the models true performance. A common practice, however, is to have a smaller separate dataset, on which models trained on the main dataset can then be retrospectively validated. The problem with this approach is, that the small dataset is usually much smaller and thus the significance of the results have to be questioned. Additionally, when predicting for a new protein, a potential model could be trained on all available data points, which is much better reflected by a leave-one-protein-out kind of cross-validation.

This validation should therefore also give a more accurate impression of what performance can actually be expected for predictions on a new protein, which was not seen before by the models that were trained, which is relevant for predicting thermo-stability increasing mutations for protein engineering.

Other validation:

An alternative to the above mentioned validation methods would be a retrospective validation, which would consists of training a model on the *T1626* dataset and then predicting for another dataset, like the *T251* dataset, which was also done by *McGuinness et al.*[17]. I too performed that retrospective validation, even on the *T96* dataset, however, since I later conducted the *leave-one-protein-out cross-validation*, which was consistent with the retrospective results from paper and my own retrospective validation, I will refrain from discussing it in this thesis, since the *leave-one-protein-out cross-validation* has the same purpose but is much more meaningful.

A final note before I move on to the description of the *3-fold bayesian optimization*: in comparison to the pre-learning, the gradient boosting models are mainly differentiated based on the features they use, since the hyper-parameters are automatically determined an can vary each run. The features used, and per

model consistent hyper-parameter choices (like training on scaled labels) are thus expressed in the model taxonomy (see section 5.1).

Hypertuning:

As previously discussed, the hypertuning was done in an automated fashion in each cross-validation split with a *3-fold bayesian optimization*[15], for which the remaining data was split three times into a one third large tune-set with the remaining data being used as a train set (see figure 6).

The bayesian optimizer was started with 15 random initial configurations in the hyper-parameter search-space. After that, 10 more configurations were iteratively tested by the bayesian optimizer, which chose the next configuration based on the *expected improvement* acquisition function. For the evaluation of a configuration, first, each configuration was evaluated independently on the three inner splits, and then the negative arithmetic mean of the calculated *MAE*s was returned, since the optimizer tries to maximize the values.

The bounds within which the optimizer had to work were the following: *{'eta': (0.01, 0.3), 'max_depth': (5, 10), 'gamma': (0.0, 1.0), colsample_bytree': (0.5, 1.0), 'subsample': (0.7, 1.0)}* which are the most common hyper-parameters to tune and their recommended ranges [9][34]. Otherwise important to note: the *max-depth* parameter was rounded down to the closest integer value before evaluation, the *Matérn* 5/2 matrix was used as the co-variance matrix, and while for the hypertuning only 100 boosting rounds were used each, the final model of each cross-validation split was trained using 1000 boosting rounds.

For the evaluation on the holdout set, the parameter combination that performed the single best in the *3-fold cross-validation* was used, to avoid unfavorable averaged hyperparameter combinations. Overall, it turned out that most often the best performing hyperparameter configurations came from the initial 15 random configurations, which indicates that either the meta-parameters of the bayesian optimization algorithms were poorly selected, or that the target function over the large five dimensional hyper-parameter space is not smooth enough for the algorithm to make meaningful guesses, which due to the relatively similar performance of many hyperparameter configurations seems probable.

[15]The *fmfn/BayesianOptimization* [4] library found at *GitHub* was used.

7 Results

In this section I will present and discuss all results related to the main objective of this thesis. I will focus on the *MAE* results for the analysis, since the *MAE* was reported in the *McGuinness et al.* [17] paper, and thus makes for a better comparison. The *RMSE* are also displayed though, for a potential comparison to other works from literature and to get a sense for the performance heavily influenced by outliers.

7.1 Cross-Validation on T1626

Merck&Co+ feature models:

These models were mainly created to establish an own baseline with the features available from the *Merck&Co+* dataset, and to reproduce results from literature (specifically the *McGuinness et al.* [17] paper) to confirm them and to potentially improve them, which are criterion (1) and (2) for evaluation success of this thesis (see section 1.3).

As we can see in the corresponding figure 9, the best performing models are the S_E_EXT ones with up to about 2.52 *MAE*, however, they are only marginally better than the S_E models (up to about 2.56 MAE), and fall within the margin of error, since the standard deviation is in the range of 0.12-0.15 *MAE* for these models. Additionally, these *EXT* features are not available for the *T251* and *T96* datasets.

Another consistency among the models is that scaling (s) the labels during training time did not have a significant effect. Additionally, the performance was only slightly worse if the filtered (f) dataset was used, but again, not significantly.

While in the case of the S_CART models, which only use the energy features from the *CART* library instead of all (E), I could exactly reproduce the results reported in the *McGuinness et al.* paper with 2.79 *MAE* (using the same features I get the same *MAE*), in the case of the S_E models I was able to improve performance from 2.67 *MAE* to 2.56 *MAE*, which corresponds to an improvement of roughly one standard deviation over the *McGuinness et al.* paper. This can probably be attributed to the more advanced architecture used (*XGBoost*) and the bayesian optimization (in the case of the paper no hypertuning was done [17]).

Extracted *CNN* feature models:

The following models are solely based on the transfer learning approach in this

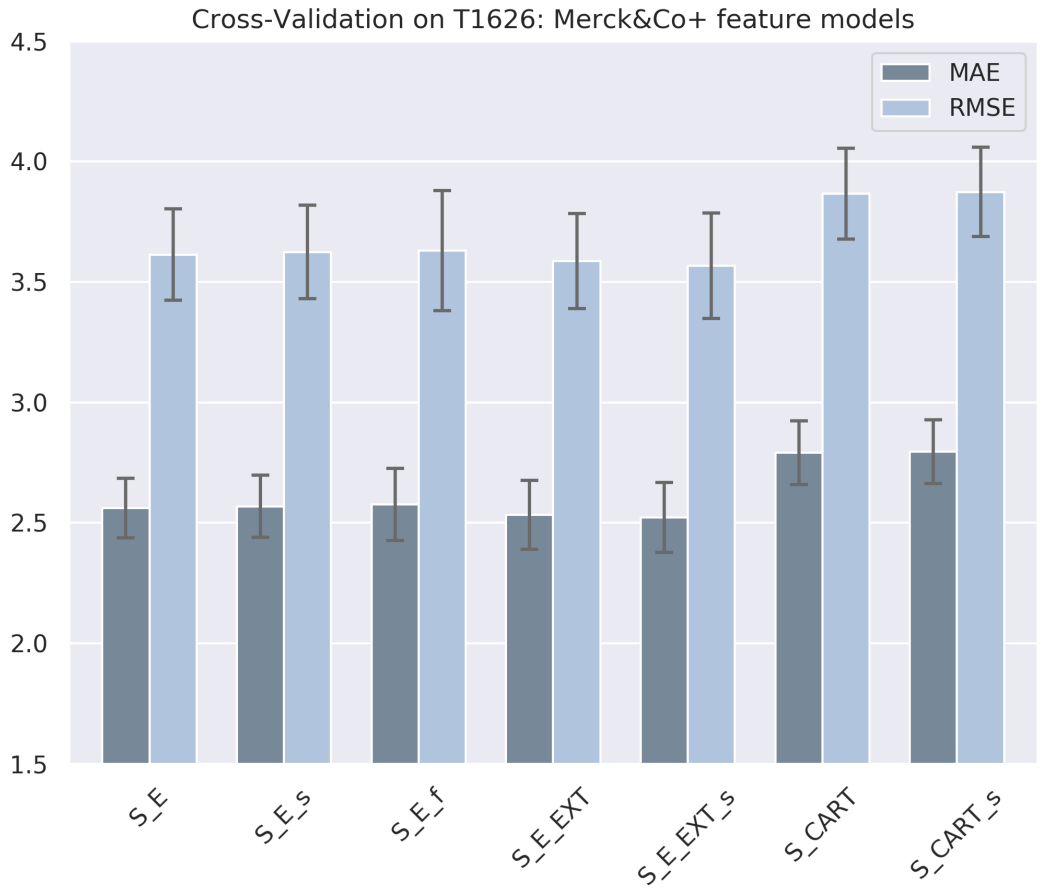

Figure 9: The error bars represent one standard deviation in the results of the cross-validation. For the exact taxonomy refer to section 5.1, for the exact numeric results to table 2 in the appendix A. Please note that the graph has been clipped from 1.5 downwards.

thesis, and thus only use the features that were extracted from the deep *CNN* model that was trained for transfer learning. The hope was to tackle success criterion (3) and (4) (see section 1.3).

The first thing to note from figure 10 is that the best model is *A3* with 3.02 *MAE*, which basically performs as good as other models that include the *A3* features, like the *A13* models.

While this is significantly worse than the best *Merck&Co+* model (by about 0.46*MAE*), it is still better than the *S* (or any subset of *S*) model from the *McGuinness et al.* paper (which achieved up to 3.09 MAE), which is quite a remarkable achievement, since the *S* feature contains not only *amino acid features* (which would be the closest feature comparable to the *AA* sequence that was used to create theses deep *CNN* extractions) but also *local* and *global* structure features, that were constructed from the *3D structure* of the protein [17]. Also, compared to *McGuinness et al.* models that were trained only on *S* features with

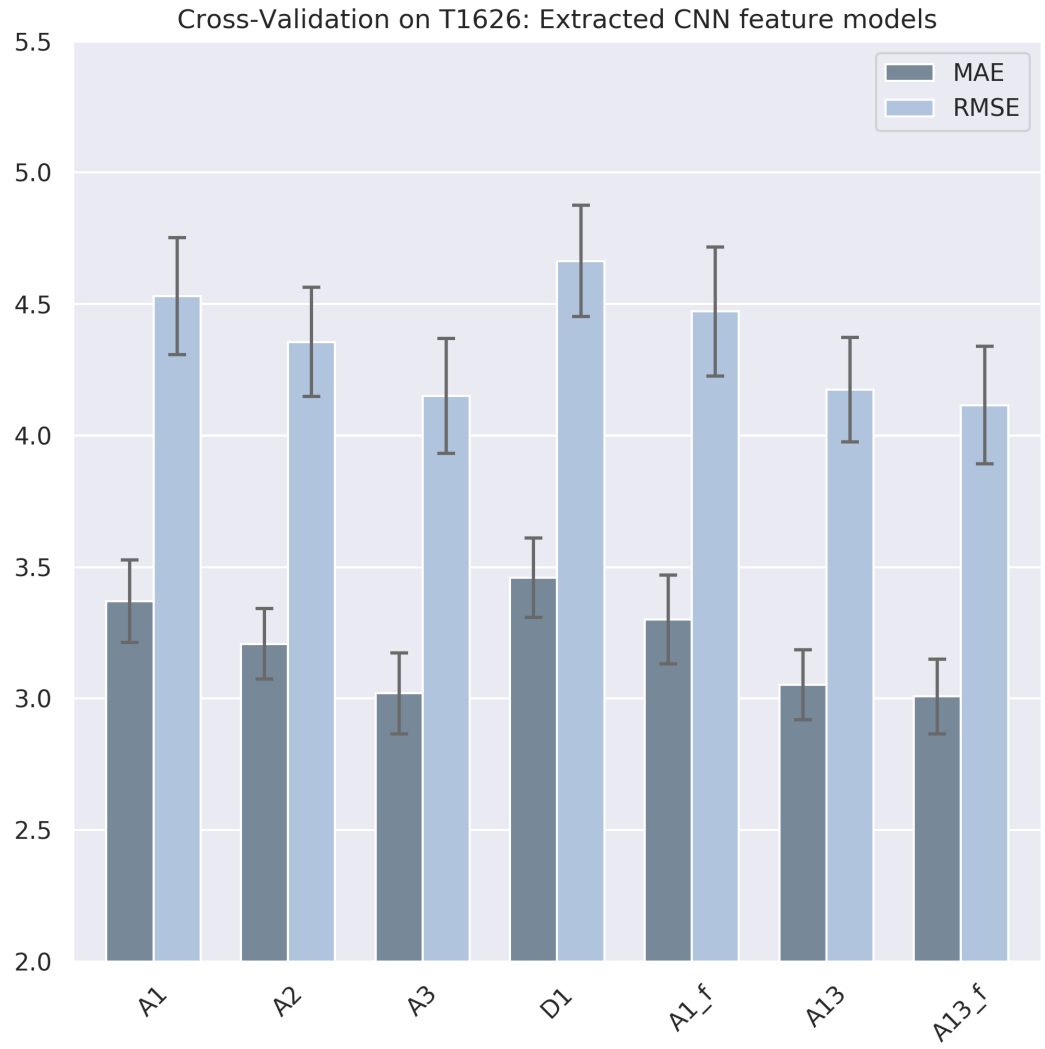

Figure 10: The error bars represent one standard deviation in the results of the cross-validation. For the exact taxonomy refer to section 5.1, for the exact numeric results to table 2 in the appendix A. Please note that the graph has been clipped from 2.0 downwards.

additional features from only one of the energy feature libraries (e.g *BL* through *MONO*) it is not much worse either, only by about 0.05-0.23 *MAE* [17].

Overall, it is only significantly outperformed by few models by *McGuinness et al.*[17] (all models that really heavily rely on energy features and the known *3D structure* of the protein) and probably[16] by *Pucci et al.* [24] (which also heavily relies on the known *3D* structure of the protein), but outperforms most other previous models. [26][28][17]

Regarding the other models depicted in figure 10 one can see a trend towards earlier extraction layers performing better, best of them of course *A3*, worst of

[16] A comparison in results is not straight forward since I did not calculate the *Pearson Correlation Coefficient*

them *A1* (see network in figure 7).

Additionally, using only the calculated difference between *wild-type* and *mutant* extractions (*D1*) seems to perform slightly worse than using the extractions additionally too (*A1*), however, only slightly. Since the performance of predicting the average on this dataset is 3.53 *MAE* [17], I deduce that the extraction layer 1 barely contains any usable information, due to the performance of the *A1* model.

Using the filtered dataset again did not change the performance in any noteworthy way.

Combined feature models:

The purpose of these models, whose performance is depicted in figure 11, was to investigate whether the extracted representations from the deep *CNN* contain any additional information that the *Merck&Co+* dataset does not.

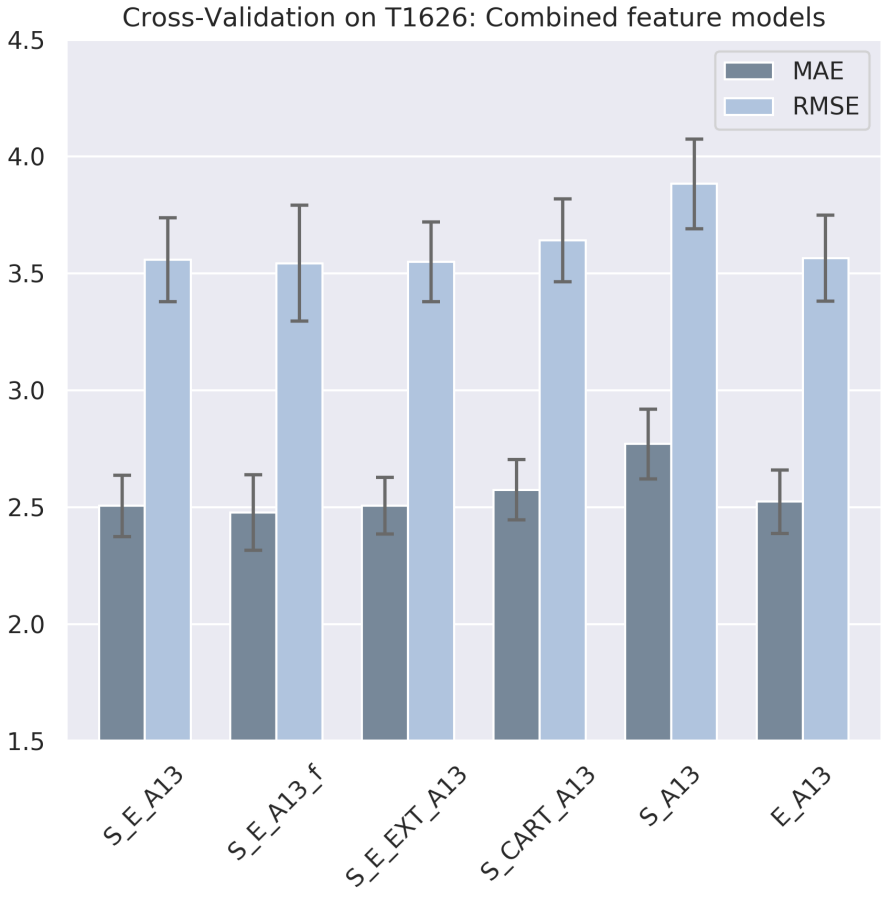

Figure 11: The error bars represent one standard deviation in the results of the cross-validation. For the exact taxonomy refer to section 5.1, for the exact numeric results to table 4 in the appendix A. Please note that the graph has been clipped from 1.5 downwards.

Unfortunately, as can be be deduced from the results, the extractions do

not contain any additional information that could lead to overall significantly better performing models. The best performing non filtered data models achieve around 2.50 *MAE* which is only a fraction of a standard deviation better than the *S_E_EXT* (2.52 *MAE*) or *S_E* (2.57 *MAE*) models on their own.

Interestingly though, the performance of the *S_CART* model was improved with the addition of *A13* from 2.79 *MAE* to 2.57 *MAE*, which is basically as good as the 2.56 *MAE* achieved with the *S_E Merck&Co+* feature model. Another surprising realization is that the extractions do not seem to be of any benefit to using all energy features (see model *E_A13*). Nevertheless, they perform well together with the *S* features, as can bee seen through *S_A13*, where performance was improved from the *A13* model by 0.28 *MAE* to 2.77 *MAE*. This is odd, since conceptually the *S* type features are much closer to the extraction features than the energy features (*E*).

Overall, the failure to provide additional value to the energy features (*E*) through the extractions can probably be attributed to the high quality of them, especially of certain *ddG* features, which leads to them dominating the models, as we will see in the next section were the feature importance of a few select models is depicted.

7.1.1 Feature Importance

As mentioned before (see section 6), the feature importance was measured using the *SHAP* library [27], in this case specifically the *TreeExplainer* for *XGBoost*. In general "SHAP values represent a feature's responsibility for a change in the model output" [27].

In the case of the upcoming *dot plots* each point represents a feature value, where red represents a high value, and blue represents a low value (relative to the distribution of values of the specific feature). The position of each point on the *x-axis* represents its magnitude and direction of impact on the output of the model. Additionally, the overall impact of a feature, as measured by the sum of its absolute *SHAP* values, is responsible for the ordering of the features on the y axis, with higher impact features being at the top of the graph. [27]

So for example a red dot situated on the very left on the *x-axis* would represent a strong negative correlation, whereas a red dot on the far right of the graph would represent a strong positive correlation of the feature value to the model output.

Merck&Co+ feature models:

From the *Merck&Co+* feature models I choose to analyse the feature importance of the **S_E** model, whose feature importance is depicted in figure 12 for

the 20 most important features.

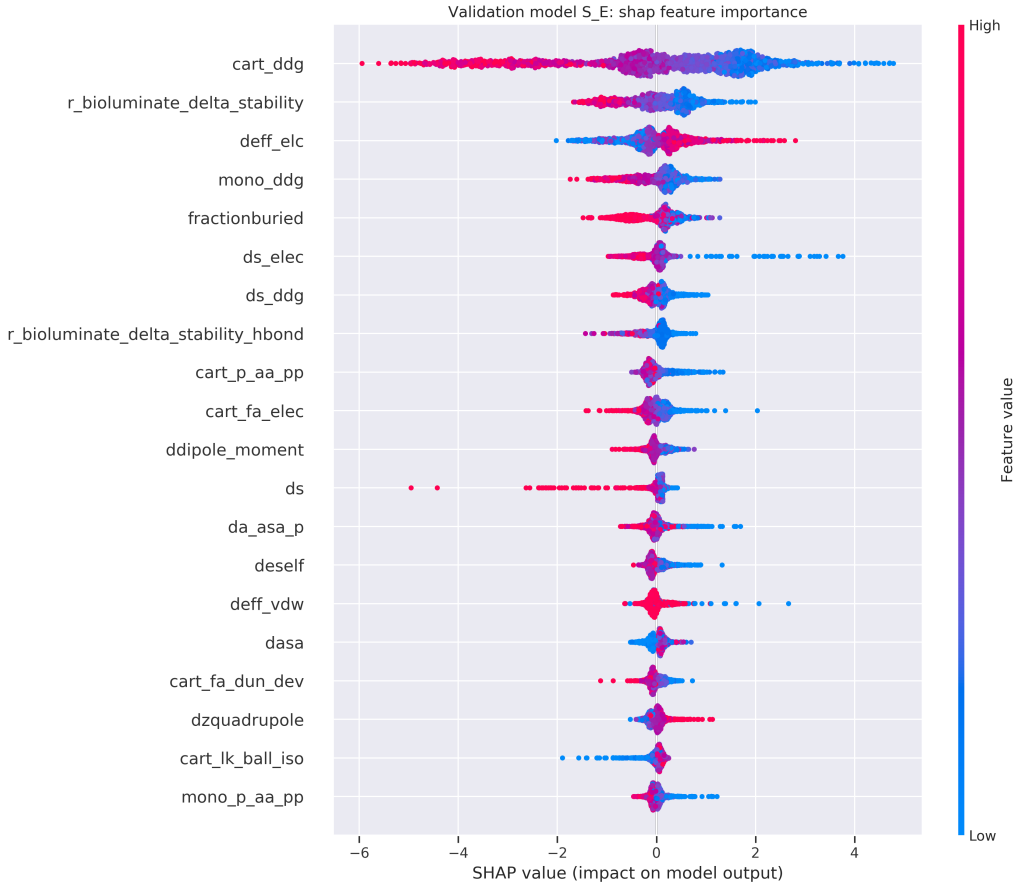

Figure 12: *SHAP TeeExplainer* output for the *S_E* model and *T1626* dataset. For the meaning of the specific features please refer to the *McGuinness et al.* [17] paper or the *Merck&Co+* dataset.

The most significant thing to notice in figure 12 is that besides the *fractionburied* feature (which is also known to significantly correlate with the melting temperature of a protein (see section 2)), the *ddipole_moment* and *dzquadrupole*, which are all three global structure features, all other features are from the energy feature category *E*.

Especially the various *ddG* features calculated by the different libraries (*cart_ddg*, *r_bioluminate_delta_stability*, *mono_ddg*, *ds_ddg*) dominate the top 20 list, most prominently among them the *cart_ddg*. This is not surprising, since it is already well known from literature that *ddG* is highly correlated with *dTm* [22][23][17]. This strong negative correlation is very clearly depicted in the graph too, since the high *ddG* values (red dots) all got great negative *SHAP* values.

Overall, most *SHAP* values are in the range of -2 to 2, except for the *cart_ddg* feature, which clearly dominates the feature importance.

Extracted *CNN* feature models:

From these models I choose to analyse the feature importance of the **A13** model, whose feature importance is depicted in figure 13 for the 20 most important features.

For the interpretation of the graph however, the nomenclature for the extraction has to be known, which unfortunately slightly differs from the model taxonomy. The name of the feature consists of the layer it was extracted from (for example *layer_ 03*) and what kind of sequence was used to create this extraction i.e. the *wild-type* sequence (*wt*), the *mutant* sequence (*mut*) or the difference of those representations (*diff*) and at what position (for example 468), since the extractions are in an array form, and a single feature corresponds to a single position in that extraction.

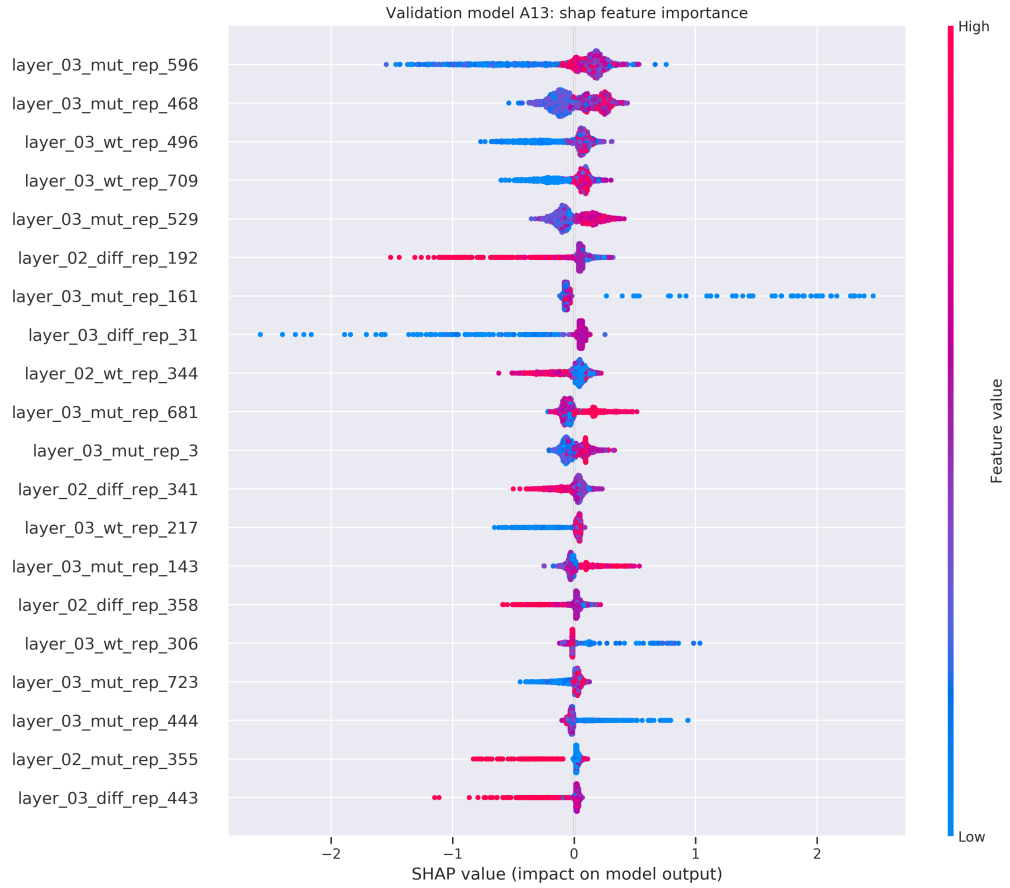

Figure 13: *SHAP TeeExplainer* output for the *A13* model and *T1626* dataset. For the meaning of the specific features please refer to the *extracted CNN feature models* section within section 4.1

The most significant observation that can be made from figure 13 is the distribution of most relevant features among the different layers: 15 of the most relevant features are from extraction layer 03, the rest from layer 02, and none from the last layer. This is consistent with the model performances depicted in figure 10. Thus it seems that extractions from the concatenation layer, which re-

ceived the least processed output from the *CNN* layers, performed the best. This might be a useful realization for further improvement using the deep transfer learning approach.

Additionally, half of the extractions are based on the mutant sequence, and the rest is split fifty-fifty among the wild-type and difference representations, which again is consistent with previous results (see figure 10).

A final realization is regarding the magnitude of the *SHAP* values: compared to the values and features depicted in figure 12, these features seam to be all of similar relevance and the *SHAP* values are mostly in the range of -1 to 1. This seems to indicate that the *CNN* feature extraction based models rely on a broader set of features in a more balanced way, compared to the *Merck&Co+* feature based models, which are heavily dominated by a few features.

Combined feature models:

For the combined feature models I choose the *S_E_A13* type model, whose feature importance is depicted in figure 14.

A surprising observation from this graph is that instead of smoothing out the relative feature importance of the individual features, the domination of certain features just escalated further. For this model almost only the first five features seem to get *SHAP* values above 1 in magnitude, and the rest almost exclusively below 1 and in most cases even below 0.5. The seemingly extreme significance of the *card_ddg* feature does not come from an even greater absolute importance, which actually seems to have decreased from the *S_E* model (see figure 12), but rather from the significantly decreased overall importance of the last 15 features that are depicted.

While there are some feature extractions represented among the top 20 features, their overall importance is dwarfed by the top five most impotent features. This may explain the miniscule improvements that were achieved by supplementing the *Merck&Co+* features with the extracted deep *CNN* features. Unfortunately, this does not give a clue as to why this happened. One explanation could be that the enormous amount of additional features that were added through the extractions also got some little feature importance each, which lead to only the very few most important features receiving a significant weight in the prediction.

7.1.2 Sanity Checks

As mentioned before, the sanity check models were constructed to test the usefulness of the extractions further, and to explore the relationship between the *wild-type* and *mutant* extractions. Since these models were not trained after the

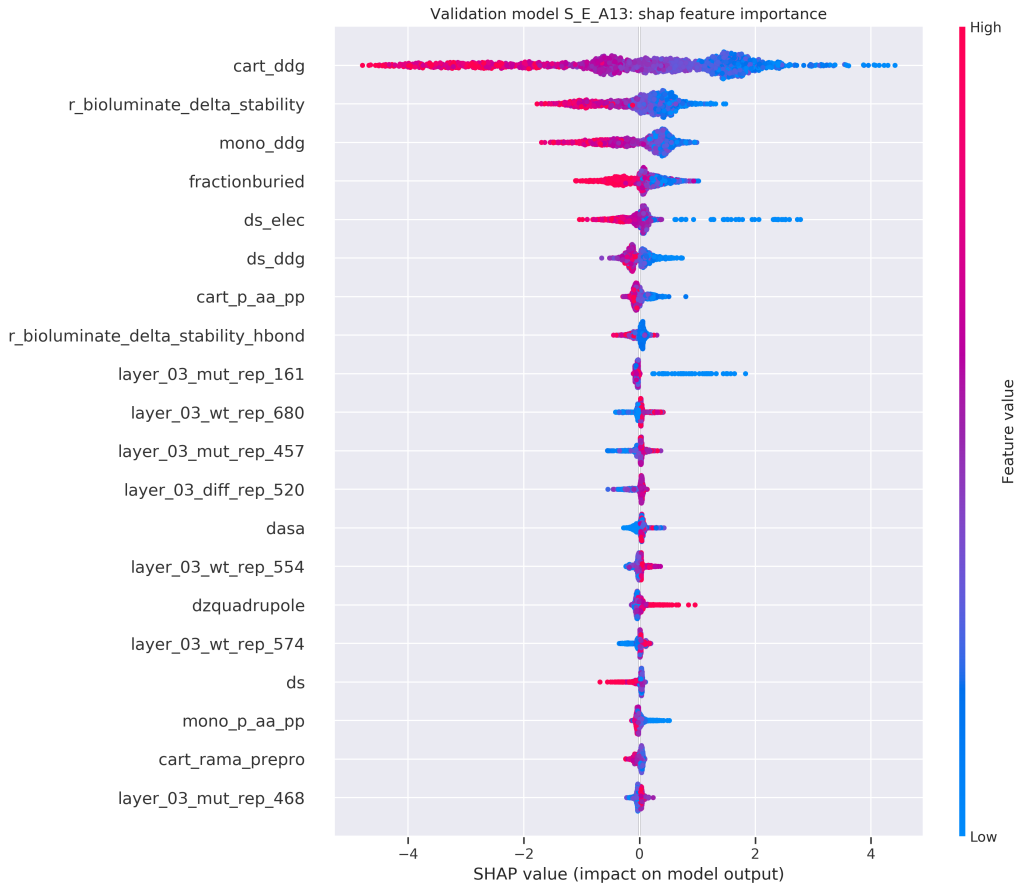

Figure 14: *SHAP TeeExplainer* output for the *S_E_A13* model and *T1626* dataset. For the meaning of the specific features please refer to the *McGuinness et al.* [17] paper or the *Merck&Co+* dataset.

other cross-validations on the *T1626* dataset, but rather in-between and along, some of the realization that follow from figure 15, which represents the results from this investigation, were already discussed through previous figures.

As you will notice in figure 15 there is a new type of model, which is marked by **raw_**. This refers to the fact that those models were not trained on the extractions that were gathered from the deep learning model, but rather on the raw (numerically encoded) amino acid sequences, as described in section 3, without any prepossessing like *one-hot encoding*[17].

As expected, the three models using only the raw sequences did not perform better than predicting the average, the *raw_W* model performed even one standard deviation worse with about 3.65 *MAE*, compared to 3.53 *MAE* [17]. While not a strong indication at all, this is additional evidence that the sequence extraction were successful and could be turned into an actually useful feature for

[17]Due to the incredibly low information density of the sequence features overall it is unlikely that a *one-hot encoding* would have given improved performance over a *numeric encoding* when using *XGBoost* with a tree booster (see discussion [18])

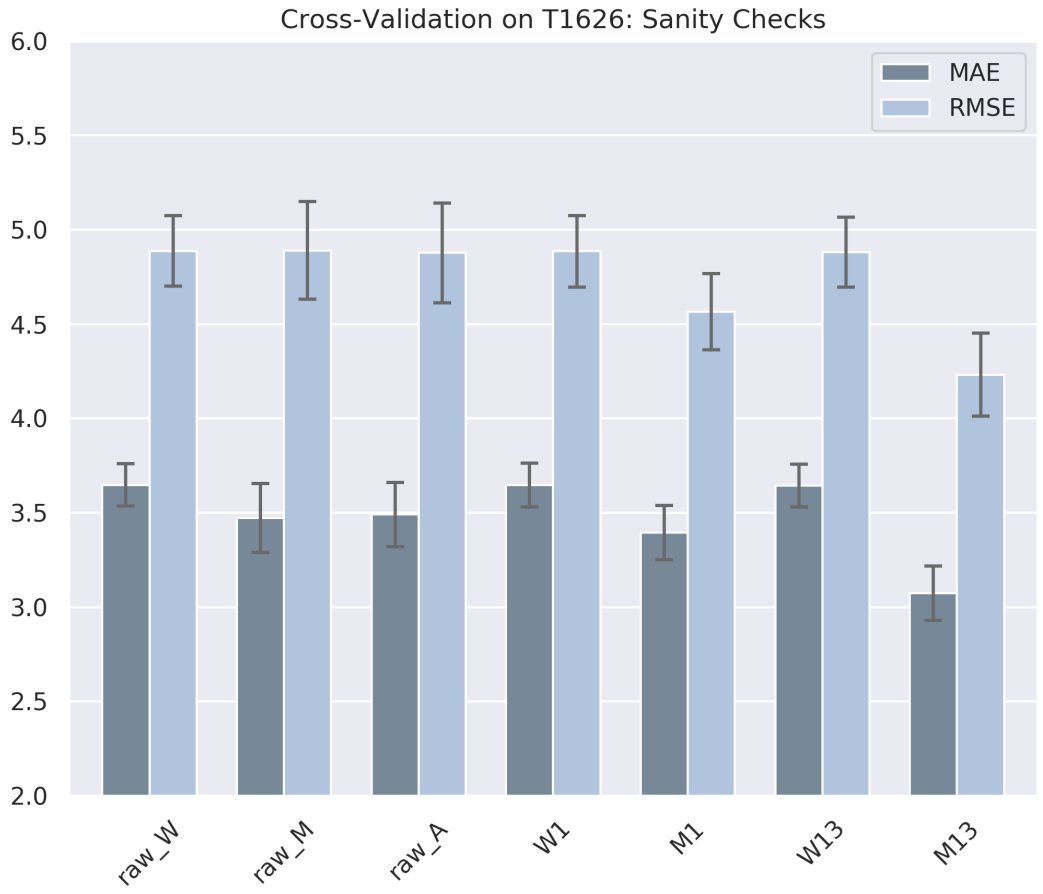

Figure 15: The error bars represent one standard deviation in the results of the cross-validation. The **raw** refers to the unprocessed, numerically encoded version of the protein *AA* sequences. For the exact taxonomy beyond that refer to section 5.1, for the exact numeric results to table 5 in the appendix A. Please note that the graph has been clipped from 2.0 downwards.

dTm prediction with the *XGBoost Tree Booster*.

The difference in performance between the *W1* and *M1* further indicates that the *mutant CNN* sequence extraction is much more important than the *wild-type CNN* sequence extraction, which is shared among all data points that have the same *wild-type* sequence. In fact *W1* performs significantly worse than predicting the average, and *M1* significantly better, however, its performance is not good overall.

In fact, as we can deduce from comparing *W13* and *M13*, the *wild-type CNN* extraction does not hold any useful information on its own, which is expected, since it is shared among all data points that contain *mutants* that are based on the same *wild-type* protein. Interestingly however, the *M13* model performs just as good as the *A13* model (3.07 *MAE* vs 3.05 *MAE*). This can probably be explained by the fact that the *mutant CNN* extraction that are based on the same *wild-type* protein probably already share some similarity that can be exploited

by the machine learning algorithm, and adding the *wild-type CNN* extractions to the features does not provide any new information.

7.2 Leave-one-protein-out Cross-Validation on T1973

So far we have dealt with the more classical kind of cross-validation and some model specific analysis, however, in order to access the performance of models more accurately, a leave-one-protein-out cross-validation was performed, which was supposed to act as a kind of proxy for a true prospective validation, which was not possible. For further explanation of the methodology refer to section 6,

Figure 16: The error bars represent one standard deviation in the results of the cross-validation, which consisted of 134 splits. For the exact taxonomy refer to section 5.1, for the exact numeric results to table 6 in the appendix A. Please note that the results displayed in this graph were weighted by the size of the holdout set (number of mutants for the left out protein), for an unweighted version refer to table 7 in appendix A

What most striking in figure 16 are the incredibly large error bars, which are roughly around 2 *MAE*. This puts all model performance means withing one standard deviation of each other, since even the biggest difference (between *M3*

and *S_E_M3*) is only about half of that. This makes it very difficult to make any meaningful observations/deductions from the graphs.

Nevertheless, a few things can be said. For example that performance of the models compared to each other seem to be distributed in the same relative way as before, however, with about 1-1.3 *MAE* less than the results from the cross-validation on the *T1626* dataset. An exception to that might be the relatively good performing average prediction model (*AVG*).

For further context: the calculated *Pearson correlation coefficient* for the *S_E_M3* model is 0.35, which represents a weak correlation, and for the *A3* model it is about 0.15, which only represents a very weak correlation. While the predictions are thus not completely uncorrelated, their usefulness has to be questioned for certain.

The drop off in performance from the more traditional cross-validation was expected, however, that it would be this stark is of course disappointing.

8 Conclusion

For the conclusion I will review the results in the context of the criteria for success that I set in the beginning of my work (see section 1.3). Additionally, I will discuss the relevance of this work by trying to discern where the value of this thesis lies withing the literature regarding the problem of predicting protein thermostability change upon single point mutation, including what prospective work that builds upon the results in this thesis could look like.

Criteria for success:

While the criteria for success that I put up in the beginning are by no means an exhaustive list of all the criteria to judge this work by, I believe they do reflect the core challenges (see section 1.2) of any new machine learning based thesis that aims to advance this particular problem domain in protein engineering.

1. *Recreate the results from literature, to create an own baseline and as a starting point:* This was covered mainly by the cross-validation performed on the *T1626* dataset (see section 7.1). In the case of a few models the performance of the models trained in this thesis matched the reported performance in the literature exactly, like is was the case for S_CART data based models. Otherwise, the results were consistent with literature too. Additionally, the feature importance analysis of the S_E model (see section 7.1.1) confirmed previous knowledge of the relevance of certain features, in particular ddG, for dTm prediction.

2. *Improve upon those results as measured by common metrics in this problem domain (like mean absolute error MAE in cross-validation), once that is done:* While I was able to improve performance significantly with some models like S_E, in most cases the improvements were marginal. Additionally, a very recent publication namely *Pucci et al.* [24] probably even outperform the S_E models significantly[18]. Also, most improvements can reasonably be explained by the more sophisticated *XGBoost* architecture together with hypertuning, and are not based on the deep *CNN* transfer learning approach. While models based on transfer learning, in particular *A13*, outperform models based on structural features (S), they do not perform better than models based on energy features (E) (see section 7).

   Nevertheless, *A13* outperforming S type based models can be regarded as a significant achievement, since the AA sequence (which are the sole feature

[18] Comparison is difficult since I did not compute the *Pearson Correlation Coefficient* for the models, and the paper surprisingly did not calculate the *MAE* or *RMSE*.

required for these models) would also fall into that category of feature, however, many S type features already rely on information extracted from the *3D structure* of the protein. In fact, this means that the *A13* model outperforms most previous models in literature, and probably represents the **best performing model in literature that does not rely on the *3D structure* of the protein**, which is not available for most proteins (about *999/1000* for which the AA sequence is known, see section 1.2).

It is also unfortunate that the combined models like *S_E_A13*, that use features from *Merck&Co+* as well as extracted features from the deep *CNN*, did not significantly improve performance. Nevertheless, some feature combinations like *S_CART_A13* seemed to complement each other well.

3. *Create models that also generalize better, meaning they also perform good in artificial prospective validation:* This criterion is mainly concerned about the decrease in performance that is seen if we move from common cross-validation to (artificial) prospective validation like leave-one-protein-out cross-validation in the case of this thesis (see section 7.2).

   Unfortunately, the models all suffer from this problem still, and the models trained with the transfer learning approach, which is supposed to improve model generalization, did not alleviate this problem.

   Additionally, the leave-one-protein-out cross-validation also shows that model performance drastically varies from protein to protein, and thus these models perform very unreliably, besides already performing worse on average than in classical cross-validation.

4. *Create models that are usable on a much larger domain, i.e. try to achieve similar performance with more widely available features:* While the transfer learning based models fit the criterion of relying only on widely available features, they do not achieve top performance overall. However, if the *3D structure* is not available for the protein in question, the *A13* model may represent the best alternative available in literature, and can probably be combined with the knowledge of a protein engineer with more domain knowledge in a productive way.

Relevance:

Regarding the protein engineering: Based on the leave-one-protein-out cross-validation I am not confident in recommending any of these models for reliably suggesting thermostability improving single point mutations, they may

only be used effectively in production as suggestions for a protein engineer to look into. Additionally, other very recent methods that are more knowledge based seem much more promising [24] in this regards, if the *3D structure* of the protein is known. Nevertheless, if the *3D structure* is not available, the *A13* model seems to be the best alternative, which could be combined with domain knowledge of the protein engineers to filter out suggested mutations that seem unrealistic, like done in the *Pucci et al.* paper [24].

Regarding future work: However, this work can be see as a proof of concept, that meaningful prediction can be made using transfer learning with the *AA* sequence alone. With more extensive and sophisticated pre-learning the current performance of models like *A13* can potentially be improved significantly. Additionally, a slightly different pre-learning approach seems very promising; instead of training on the whole sequence of a protein, only train on a fixed size *AA* window (for example 31 residues wide) for pre-learning, that can then be applied to the position of the single point mutation for feature extraction[19], since the effect of single point mutations is mostly local in the case of useful mutations anyways. And finally, approaches based solely on the *AA* sequence can be combined with pipelines to remove insensible suggestions, that would interfere with the protein function or are just unlikely from an evolutionary stance point, like it is done in the *Pucci et al.* paper [24], to further improve performance.

Additionally, I think this work further supports the thesis that it is not the lack of sophisticated machine learning techniques that limits performance, but rather the lack of data, as expressed many times previously [17]. This can only be compensated with more data, or a more domain knowledge based approach with a deeper understanding of the correlations between protein structure and melting temperature, which seems to be the more promising approach [24].

[19] As suggested by Prof. Dr. Dirk Walther

9 Acknowledgements

I would like to thank the *Machine Learning Group* at *University of Potsdam* for their great machine learning insight and support, especially *Prof. Dr. Tobias Scheffer* and *Dr. Paul Prasse* for supervising this thesis.

Also great thanks to the *iGEM Potsdam 2019* team whose project was the inspiration for this thesis. I would like to thank them for their previous work, biological insight and support, especially *Lukas Golombek*, *Jonas Kopka* and *Bryan Nowack* for their critical reflections.

A Appendix

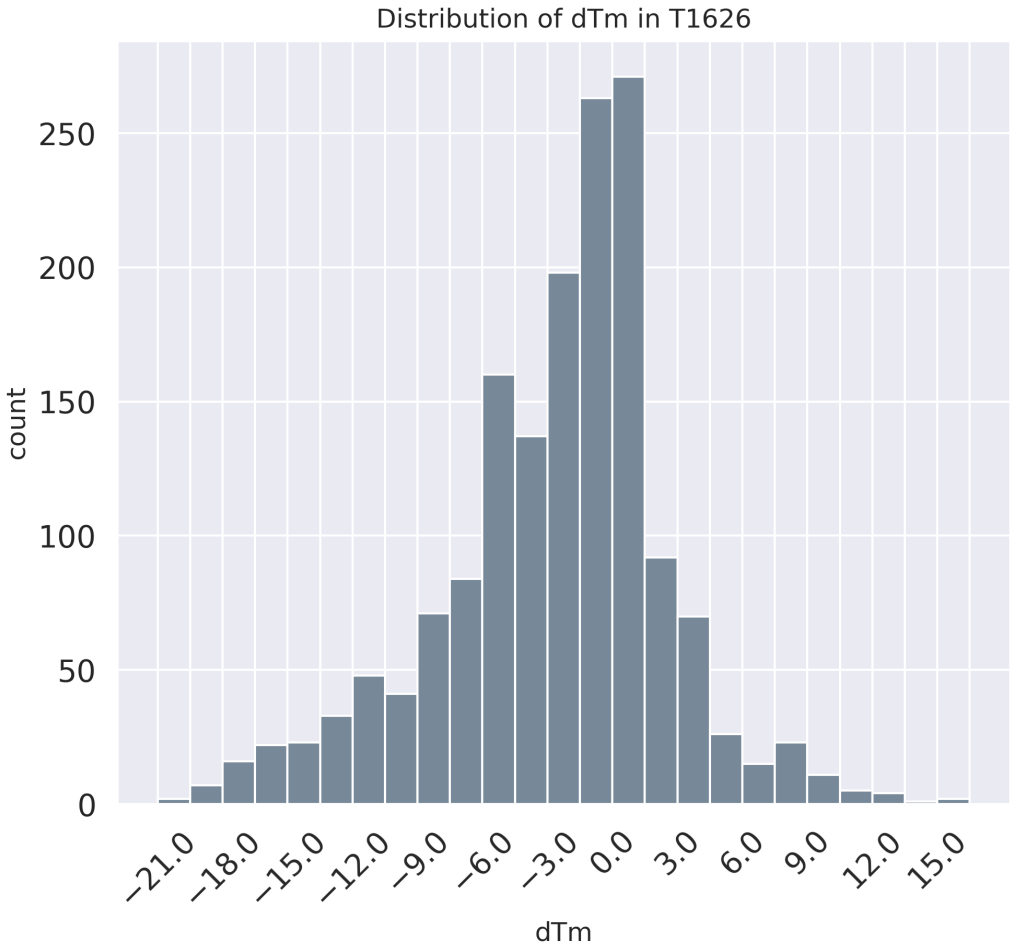

Figure 17: The distribution of change in melting temperature (dTm) in $°C$ between wild-type and mutant sequence in the *T1626* dataset.

	MAE	RMSE	MAE_std	RMSE_std
avg_pred	8.913	12.696		
default_t	6.252	8.93	0.068	0.13
default_h	6.241	8.912	0.07	0.128
best_t	5.718	8.391	0.169	0.111
best_h	5.709	8.379	0.167	0.107
best_scale+_t	5.709	8.379	0.167	0.108
train+	5.454	8.091		

Table 1: Performance of different models on BacDive+

	MAE	RMSE	MAE_std	RMSE_std
S_E	2.562	3.614	0.124	0.189
S_E_s	2.569	3.625	0.129	0.194
S_E_f	2.577	3.63	0.15	0.25
S_E_EXT	2.533	3.586	0.143	0.197
S_E_EXT_s	2.522	3.567	0.144	0.219
S_CART	2.791	3.867	0.132	0.189
S_CART_s	2.796	3.874	0.132	0.185

Table 2: Cross-Validation performance on *T1626*: *Merck&Co* feature models.

	MAE	RMSE	MAE_std	RMSE_std
A1	3.37	4.529	0.156	0.223
A2	3.208	4.356	0.134	0.207
A3	3.021	4.151	0.154	0.219
D1	3.46	4.664	0.15	0.212
A1_f	3.301	4.472	0.168	0.245
A13	3.052	4.175	0.133	0.198
A13_f	3.008	4.116	0.142	0.223

Table 3: Cross-Validation performance on *T1626*: Extracted *CNN* feature models.

	MAE	RMSE	MAE_std	RMSE_std
S_E_A13	2.505	3.558	0.13	0.18
S_E_A13_f	2.477	3.543	0.161	0.247
S_E_EXT_A13	2.505	3.548	0.122	0.171
S_CART_A13	2.574	3.64	0.13	0.177
S_A13	2.769	3.882	0.149	0.192
E_A13	2.523	3.564	0.136	0.183

Table 4: Cross-Validation performance on *T1626*: Combined feature models.

	MAE	RMSE	MAE_std	RMSE_std
raw_W	3.648	4.888	0.111	0.187
raw_M	3.472	4.891	0.182	0.258
raw_A	3.49	4.878	0.17	0.264
W1	3.647	4.885	0.115	0.189
M1	3.395	4.565	0.144	0.203
W13	3.643	4.882	0.114	0.186
M13	3.074	4.232	0.143	0.22

Table 5: Cross-Validation performance on *T1626*: Sanity Checks

	MAE	RMSE	MAE_std	RMSE_std
S_E	3.613	4.483	1.854	1.993
A3	4.235	5.105	1.96	2.134
M3	4.338	5.22	2.104	2.238
S_E_M3	3.449	4.241	1.792	1.943
S	4.091	5.01	1.931	2.029
S_CART	3.798	4.687	1.874	1.956
AVG	4.397	5.256	1.819	1.993

Table 6: Weighted leave-one-protein-out cross-validation on T1973.

	MAE	RMSE	MAE_std	RMSE_std
S_E	4.037	4.652	2.6	2.787
A3	4.713	5.258	2.906	3.024
M3	4.827	5.379	3.266	3.36
S_E_M3	4.102	4.681	2.641	2.807
S	4.579	5.172	2.665	2.731
S_CART	4.21	4.811	2.586	2.734
AVG	4.909	5.417	2.853	2.946

Table 7: Leave-one-protein-out cross-validation on T1973.

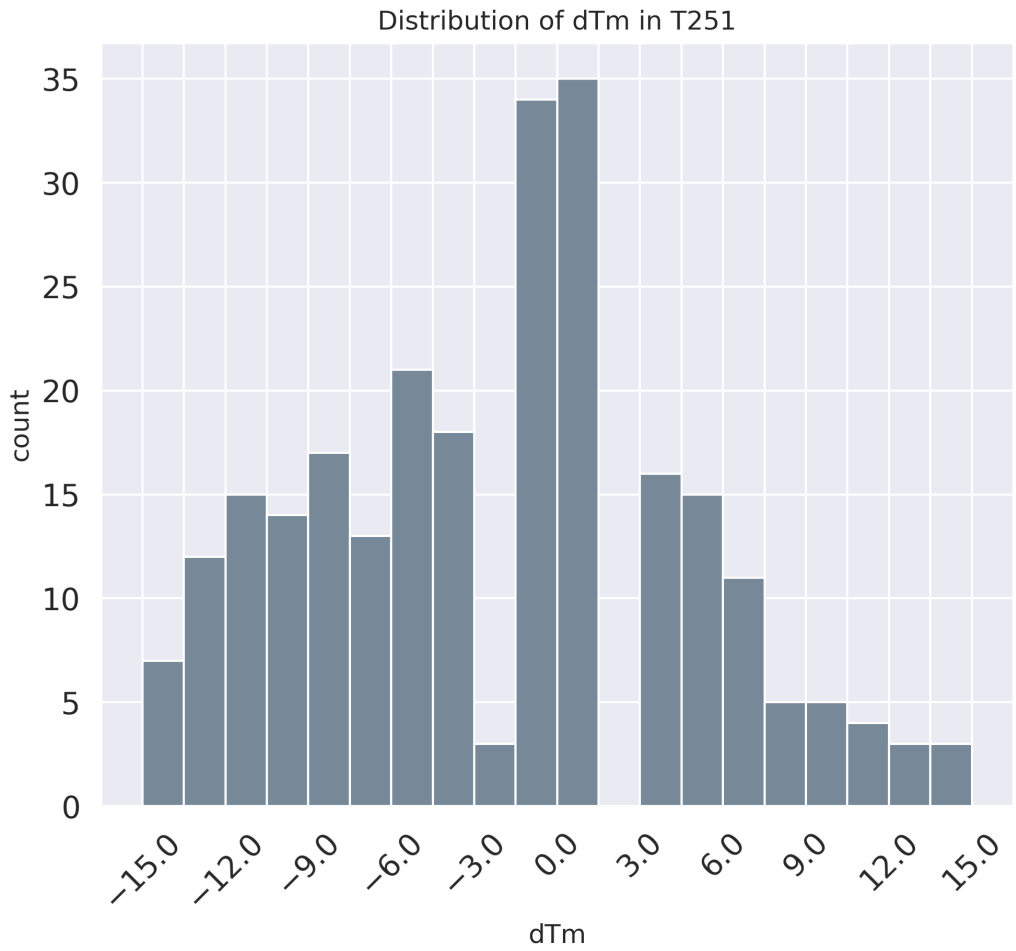

Figure 18: The distribution of change in melting temperature (dTm) in $°C$ between wild-type and mutant sequence in the *T251* dataset.

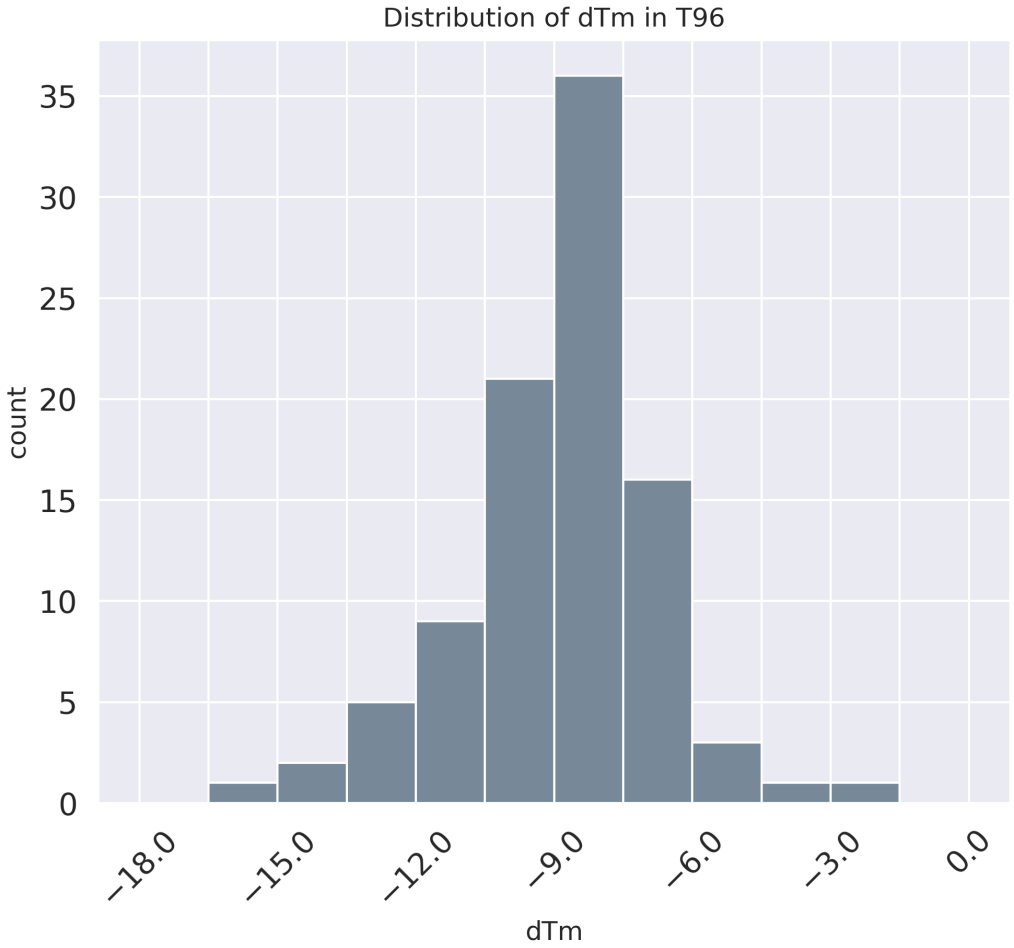

Figure 19: The distribution of change in melting temperature (dTm) in $°C$ between wild-type and mutant sequence in the *T96* dataset.

```
inputs = tf.keras.layers.Input(shape=(SEQUENCE_LEN,))

x = tf.keras.layers.Embedding(CLASSES, 8,
input_length=SEQUENCE_LEN)(inputs)

x = tf.keras.layers.Conv1D(128, 7, 2)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)
x = tf.keras.layers.Conv1D(128, 3)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)
x = tf.keras.layers.Conv1D(128, 3)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)

x = tf.keras.layers.MaxPooling1D(3)(x)
x = tf.keras.layers.Conv1D(256, 3)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)
x = tf.keras.layers.Conv1D(256, 3)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)
x = tf.keras.layers.Conv1D(256, 3)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)

x = tf.keras.layers.GlobalAveragePooling1D()(x)
x = tf.keras.layers.Dense(256)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation("relu")(x)
x = tf.keras.layers.Dense(1, activation="linear")(x)

model = tf.keras.Model(inputs=inputs, outputs=x)
```

Figure 20: The baseline artificial neural network that was later improved by hyper-tuning, as it was defined using *Keras*.

References

[1] AzaToth. Myoglobin. https://en.wikipedia.org/wiki/Enzyme#/media/File:Myoglobin.png, Feb 2008.

[2] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 01 2000.

[3] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*, Jun 2016.

[4] Fmfn. fmfn/bayesianoptimization. https://github.com/fmfn/BayesianOptimization, Aug 2019.

[5] Indrayudh Ghosal and Giles Hooker. Boosting Random Forests to Reduce Bias; One-Step Boosted Forest and its Variance Estimate. *arXiv e-prints*, page arXiv:1803.08000, Mar 2018.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2016. page 526.

[7] Google. Google Colab. https://colab.research.google.com/, 2019. [Online; accessed 15-August-2019].

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, Dec 2015.

[9] Aarshay Jain. Complete guide to parameter tuning in xgboost (with codes in python). https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/, Aug 2019.

[10] Lei Jia, Ramya Yarlagadda, and Charles C. Reed. Structure based thermostability prediction models for protein single point mutations with machine learning tools. *Plos One*, 10(9), 2015.

[11] Norman Jouppi, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb,

Cliff Young, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C Richard Ho, Doug Hogberg, John Hu, and Nan Boden. In-datacenter performance analysis of a tensor processing unit. In *Conference Paper*, pages 1–12, 06 2017.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[13] M. D. S. Kumar, K. A. Bava, M. M. Gromiha, P. Prabakaran, K. Kitajima, H. Uedaira, and A. Sarai. Protherm and pronit: thermodynamic databases for proteins and protein-nucleic acid interactions. *Nucleic Acids Research*, 34(90001), 2006.

[14] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1701.04128, Jan 2017.

[16] M. Masso and I. I. Vaisman. Accurate prediction of stability changes in protein mutants by combining machine learning with structure based computational mutagenesis. *Bioinformatics*, 24(18):2002–2009, 2008.

[17] Kenneth N. Mcguinness, Weilan Pan, Robert P. Sheridan, Grant Murphy, and Alejandro Crespo. Role of simple descriptors and applicability domain in predicting change in protein thermostability. *Plos One*, 13(9), 2018.

[18] Miscellaneous. Why one-hot-encoding gives worse scores? `https://www.kaggle.com/c/zillow-prize-1/discussion/38793`.

[19] Hassan Pezeshgi Modarres, Mohammad R. Mofrad, and Amir Sanati-Nezhad. Protdatatherm: A database for thermostability analysis and engineering of proteins. *Plos One*, 13(1), 2018.

[20] Nuala A. Oleary, Mathew W. Wright, J. Rodney Brister, Stacy Ciufo, Diana Haddad, Rich Mcveigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, and et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research*, 44(D1), 2015.

[21] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *arXiv e-prints*, page arXiv:1804.02763, Apr 2018.

[22] Fabrizio Pucci, Raphaël Bourgeas, and Marianne Rooman. Predicting protein thermal stability changes upon point mutations using statistical potentials: Introducing hotmusic. *Scientific Reports*, 6(1), 2016.

[23] Fabrizio Pucci, Malik Dhanani, Yves Dehouck, and Marianne Rooman. Protein thermostability prediction within homologous families using temperature-dependent statistical potentials. *PLoS ONE*, 9(3), 2014.

[24] Fabrizio Pucci, Jean Marc Kwasigroch, and Marianne Rooman. Protein thermal stability engineering using hotmusic. *bioRxiv*, 2019.

[25] Lorenz Christian Reimer, Anna Vetcininova, Joaquim Sardà Carbasse, Carola Söhngen, Dorothea Gleim, Christian Ebeling, and Jörg Overmann. Bacdive in 2019: bacterial phenotypic data for high-throughput biodiversity analysis. *Nucleic Acids Research*, 47(D1), 2018.

[26] K. Saraboji, M. Michael Gromiha, and M. N. Ponnuswamy. Average assignment method for predicting the stability of protein mutants. *Biopolymers*, 82(1):80–92, May 2006.

[27] Slundberg. slundberg/shap. `https://github.com/slundberg/shap`, Aug 2019.

[28] C. M. Topham, N. Srinivasan, and T. L. Blundell. Prediction of the stability of protein mutants based on structural environment-dependent amino acid substitution and propensity tables. *Protein Engineering Design and Selection*, 10(1):7–21, 1997.

[29] Wikipedia. Artificial neural network — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=908473727`, 2019. [Online; accessed 07-August-2019].

[30] Wikipedia. Decision tree learning — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Decision%20tree%20learning&oldid=908472869`, 2019. [Online; accessed 18-August-2019].

[31] Wikipedia. Protein — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Protein&oldid=904032385`, 2019. [Online; accessed 10-July-2019].

[32] Wikipedia. Thermostability — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Thermostability&oldid=904669855, 2019. [Online; accessed 07-August-2019].

[33] Wikipedia. Transfer learning — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Transfer%20learning&oldid=909759697, 2019. [Online; accessed 07-August-2019].

[34] xgboost developers. Xgboost parameters. https://xgboost.readthedocs.io/en/latest/parameter.html.

[35] Kevin K. Yang, Zong wei Wu, and Frances H. Arnold. Machine learning in protein engineering. *arXiv e-prints*, 2018.

[36] Javad Zahiri. An overview of the protein thermostability prediction: Databases and tools. *Journal of Nanomedicine Research*, 3(6), 2016.

[37] Yong-Chun Zuo, Guo-Liang Fan, Wei Chen, and Qian-Zhong Li. A similarity distance of diversity measure for discriminating mesophilic and thermophilic proteins, Aug 2012.