TREATING BIOLOGICAL SEQUENCES AS NATURAL LANGUAGE, A CASE STUDY ON SUB-CELLULAR PROTEIN LOCALIZATION

A Thesis by

Jarret Ross

Bachelor of Science, Wichita State University, 2013

Submitted to the Department of Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

May 2016

TREATING BIOLOGICAL SEQUENCES AS NATURAL LANGUAGE, A CASE STUDY ON SUB-CELLULAR PROTEIN LOCALIZATION

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Computer Science.

Kaushik Sinha, Committee Chair

Huzefa Kagdi, Committee Member

Holger Meyer, Committee Member

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Chapter **Page**

List of Figures

List of Tables

Table **Page**

ABSTRACT

Extracting meaning out of biological sequences such as DNA, RNA, and strings of amino acids is a task that traditionally requires a large amount of expert knowledge. Breakthroughs and advancements of these subjects are slow due to the computational intractability inherent in biological sequences. If it were possible to lower or remove the high level of expertise needed to solve important problems in biology it might be possible to increase the pace of biological breakthroughs. As a small step in this direction this thesis focuses on the challenge of sub-cellular protein localization. It is possible to totally remove the need for any biological understanding by viewing the problem of Sub-cellular protein localization as a Natural Language Processing task. This method requires no hand engineered features and performs at a character level granularity. Modifications are made to an existing deep convolution network which was designed to perform a range of Natural Language Processing tasks such as Sentiment Analysis and Topic Classification. While this model does not achieve state of the art performance it is competitive with respect to other models evaluated in this Thesis. These findings are encouraging for a few reasons. First it is shown that a totally biologically naive method performs competitively with other hand engineered methods. Lastly it is hoped that the current intense research focus on Natural Language processing in the field of deep learning will greatly increase the viability of the method contained in this thesis in coming years.

Chapter 1

INTRODUCTION AND FOUNDATIONAL WORKS

1.1 Introduction

Biological cells are composed of several proteins of which each protein belongs to some family. The family a protein belongs to is referred to its a Sub-cellular location or Sub-cellular compartment. Each Sub-cellular protein location has specific properties that are dictated by biochemical micro-environments. Properties of each sub-cellular protein location determine such things as how the protein will interact with other proteins, what function the protein will perform in the cell as well as the potential rolls the cell containing the protein will play.

Being able to identify, or annotate, the sub-cellular location of unknown proteins is a key goal of the Proteomics discipline. Previous methods of identifying which Sub-cellular location a protein belongs to, also known as Sub-cellular protein localization, range from laboratory experimentation to computation methods. The problems with all of the previous methods is that they require a various range of biological knowledge to perform the task of Sub-cellular protein localization. This overhead of knowledge needed to perform Sub-cellular protein localization limits the speed of potential discovery due to the limited number of people who

can perform the task.

Table 1.1: All Characters of the Amino Acid Alphabet

ACEDGFIHKMLNQPSRTWVY

As an attempt to get around the knowledge overhead required to perform Sub-cellular protein localization this thesis frames the task of sub-cellular protein localization as a Natural Language Processing (NLP) task. The impetus is to reduce the amount of domain knowledge needed to perform classification tasks on biological sequences, in this case proteins, down to zero. The intuition that viewing the problem of Sub-cellular protein localization as a natural language processing task is reasonable due to the fact that proteins consists of sequences of amino acids. Amino acids can be represented by characters which compose an alphabet smaller than the English alphabet. Sequences of amino acids can be viewed as strings of characters, which are a fundamental object of natural language. Thus Sub-cellular protein localization reduces down to a natural language problem. Two example of a proteins and there amino acid string representation are presented below and the alphabet for the all amino acids contained in the data set focused on in this thesis can be found in table 1.1.

Protein example from the Plasma Membrane and Nuclear Sub-cellular locations

- `MENQPVRWRALPGLPRPPGLPAAPWLLLGVLLLPGTLRLAGGQSVTHTGLPIMASLANTAISFSCRITYPY TPQFKVFTVSYFHEDLQGQRSPKKPTNCHPGLGTENQSHTLDCQVTLVLPGASATGTYYCSVHWPHSTVRG SGTFILVRDAGYREPPQSPQKLLLFGFTGLLSVLSVVGTALLLWNKKRMRGPGKDPTRKCPDPRSASSPKQ HPSESVYTALQRRETEVYACIENEDGSSPTAKQSPLSQERPHRFEDDGELNLVYENL`

- `METAGKAKKGFGGRKGGPRKKSVTRSVKAGLQFPVGRIGRYLKKGRYAQRVGTGAPVYLAAVLEYLAAEVL ELAGNAARDNKKTRIIPRHLLLAVRNDEELGKLLAGVTFAHGGVLPNINPVLLPKKTAEKAAKEPKSPSKA GKSPKKA`

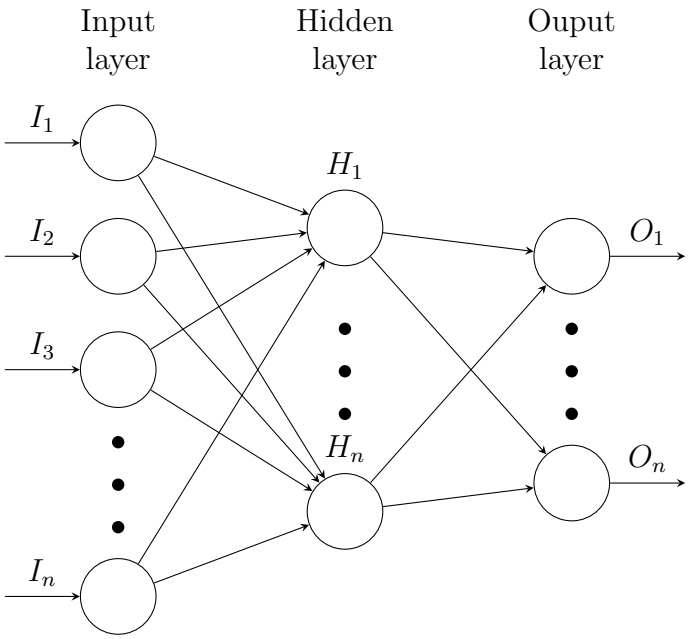


Figure 1.1: An Example of a Fully Connected Multi-Layer Perceptron

Looking to the future, by viewing biological sequences as natural language it will be possible to ride the coattails of current and future NLP advancements brought about by the intense research interest the subject is experiencing in the Deep Learning discipline. This means extracting meaning from biological sequences will not be limited to any certain architecture and will be able to benefit from any NLP advancement.

Experiments are performed using a Character Level Text Classification network the was modified slightly for the sub cellular protein localization task. Comparisons are made between the results of the Character Level Text Classification model and others ranging from a heavily hand engineered model to a more biologically naive implementation which still use elements from substitution matrices as well as protein profiles to enrich its inputs. Only single model systems are focused on for all of comparisons.

The details of the over fitting behaviour of three slight variations of the model of focus in this work are discussed. Graphs of the objective functions of each variation are used to analyse their respective over fitting behaviour.

Potential problems of the model in this thesis are discussed as well as a look to future

work that has the potential to over come the challenges faced in the current implementation.

This thesis is broken into five chapters. The rest of this chapter discusses previous work that serves as the foundation for the work in this thesis. Chapter 2 described the modules used to build the neural network evaluated in this work, the data set which experiments are evaluated on and how amino acid strings are processed into an input for the network. Chapter 3 discusses the experimental set-up, results and insights gained from the experimentation. Chapter 4 discusses problems experienced with the methods that are evaluated in this thesis as well as possible avenues for future work. Finally chapter 5 concludes this thesis.

1.2 Previous Work

1.2.1 Character Level Text Classification

In [**?**] the authors introduce a convolution neural network that performs the task of character-level text classification. Traditionally text classification has been done at the word level and has been accomplished utilizing various methods. Methods for word level text classification range from counting techniques, such as n-grams with TF-IDF [**?**], to neural network implementations, such as Long Short Term Memory (LSTM) [**?**] networks as well as Convolution networks [**?**].

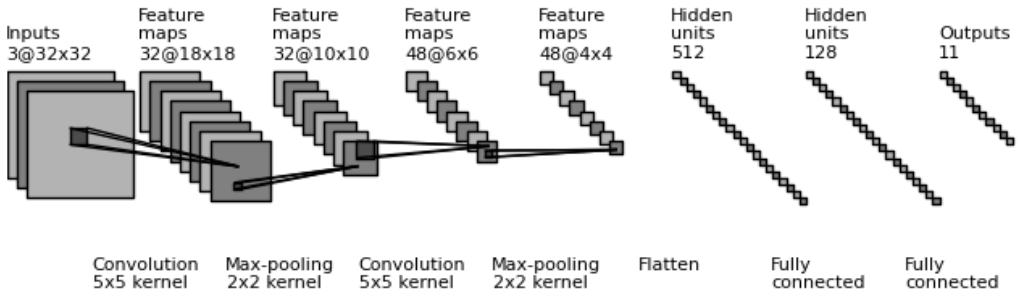


Figure 1.2: Example of a Spatial Convolution Neural Network

The character level convolution implementation has the benefit of not needing knowledge of words or knowledge of syntactic or semantic structure of a language. This lack of structural knowledge requirements is exactly what is needed to reduces the amount of domain knowledge for performing Sub-cellular protein localization down to zero.

The Character Level Text Classification models alphabet consists of 70 characters with 26 letters of the English alphabet, in only one case, 10 digits and 33 other characters and the new line character. The models performance is examined on several tasks.

It is shown to perform well on the task of sentiment analysis on Amazon reviews. In this experiment an Amazon review is treated as the input and the model has to determine which of the five sentiment classes, ranging from negative to positive, the review belongs to.

The model is also shown to perform well on the task of News categorization in English as well as in Chinese. The Chinese articles are Latinzed and the same alphabet is used for both English and Chinese articles. In these experiments an article was fed into the network which then determined if the article belonged to either the 'Sports', 'Finance', 'Entertainment', 'Automobile' or 'Technology' class for the Chinese test. For the English test the categories consisted of 'World', 'Sports', 'Business' and 'Sci/Tech'.

Tasks such as Topic Classification on the Yahoo! Answers Comprehensive Questions and Answers dataset and Ontology Classification on the DBpedia knowledge base were also performed.

The ability of this network to perform varied tasks on multiple natural languages while operating on the character level was ultimately the motivation to use it to experiment on the task of pre-cellular protein localization.

1.2.2 Sub-Cellular Localization Models

A comparison is made between this single model developed for this thesis and inspired by the above network to other single models on the same data set. The comparison models range from a heavily hand engineered model that requires a deep understanding of biology to a

model that mixes a convolution layer whose output is feed into an LSTM. A brief description of both comparison models follows.

The heavily hand engineered model is termed MultiLoc [**?**]. MulitLoc takes as input a query sequence which is feed into four modules. These modules are termed SVMTarget, SVMaac, SVMSA and MotifSearch. The output of all of these sub-models are stored into a structure termed the protein profile vector (PPV) which is feed into a one-versus-one support vector machine (SVM) which then outputs a classification.

As an example of how deep an individuals knowledge of biology needs to be to build such a system a direct quote of the description of one of the MultiLoc modules follows. "SVMTarget predicts localization categories based on N-terminal targeting sequences. The plant version predicts four categories based on the type of targeting sequence: chloroplast transit peptides, mitochondrial transit peptides, targeting peptides of the secretory pathway and other proteins lacking N-terminal targeting sequences. " It is apparent that a deep biological knowledge is needed just to understand a small section of a description of a single small element of the MultiLoc model.

Lastly [**?**] experiments with three models all of which use a single convolution layer without pooling as a feature extractor. The output of the convolution layer is then fed into a bidirectional Long Short-term Memory (LSTM) recurrent neural network. These models differ in that one uses an attention mechanism, termed A-LSTM, as well as the LSTM while the other model does not utilize an attention mechanism and relies only on a bidirectional LSTM and is termed R-LSTM. The third model is actually an ensemble of the authors models. Since we are focusing on single model implementation the details of the ensemble model are omitted.

An important aspect of the method in [**?**] is how the amino acid character strings are converted into network friendly inputs. This conversion is done by converting each amino acid character in the sequence into the concatenation of a 1-of-k representation of the individual

amino acid. Next a protein profile created by ProfilePro[1] is concatenated to the 1-of-k representation. Concatenated to that resulting vector is the column associated with the individual amino acid from the Blosum80 [**?**] substitution matrix and the column associated with the HSDM [**?**] substitution matrix. This means that each amino acid character is mapped into a numerical vector of length 80.

The authors also truncate the input amino acids sequence by removing the center of the amino acid. The rational for this method of truncation is because of the N-terminal properties absent in the center of the amino acid.

In this work the view taken is that the use of the the substitution matrix columns and protein profile as well of the rational for truncation of the center of the sequence to constitute domain knowledge and potentially hand engineering (the amount of work that went into the creation of the substitution matrices as well as protein profilers was no doubt extensive).

In this work all hand engineering and domain knowledge is removed by just treating the input as natural language and the task as a text classification task.

[1] `http://download.igb.uci.edu/`

Chapter 2

MODEL DETAILS AND DATA SET

2.1 Foundational Modules

The network in this work is composed of six major components. These components are as follows; Temporal convolution [**?**], Temporal max pooling [**?**, **?**, **?**], a non-linearity similiar to Rectified Linear unites [**?**] termed Threshold , Dropout [**?**] for regularization, Linear Layers and a LogSoftmax Layer [**?**]. A brief description of each model is given in this section. As a note, the term Temporal indicates that the pooling on convolution is done in one dimension rather than the more popular two dimension applications also referred to as spatial applications.

2.1.1 Temporal Convolution

Temporal Convolution [**?**] is a convolution on a 1-dimension input. From an abstract point of view a convolution acts as a feature extractor given the input that the convolution operates on. The sophistication of the feature extracted by the convolution gets more elaborate as more layers of convolutions are stacked on each other. As an analogy, from visual recognition, given a stack of convolution layers the convolutions at the bottom of the stack tend to learn edges and other fundamental shapes while the convolution layers higher up in the stack tend to assemble the more fundamental features learned from the bottom layer to create features

as complex as faces, animals and other complex objects. While convolution on images is a 2 dimensional problem the property of learning increasingly elaborate features still holds in the 1 dimensional case. A more mathematical definition of convolution follows.

Given an input vector $X \in \mathbb{R}^p$ where $X = [x_1, \ldots, x_p]$ and a kernel consisting of learned weights $W \in \mathbb{R}^n$ defined as $W = [w_1, w_2, \ldots, w_n]$ where $k \geq n$. A convolution $g(X)$ is defined by equation 2.1 which shows the computation of an arbitrary index of the output of the convolution.

$$g(X)_i = \Sigma_{k=n}^{1} x_{i*d-k+c} * w_k \tag{2.1}$$

The index of the convolution output is designated by i and d is the stride of the convolution. The stride is defined as how far the kernel slides down the input vector at each convolution step. Lastly $c = n - d + 1$ where n is the number of weights in the convolution kernel.

2.1.2 Temporal Max Pooling

Another module termed Temporal Max Pooling [**?**,**?**,**?**] is much simpler than the convolution layer. Max Pooling shares the concept of stride and kernel with convolutions. While the stride is identical to that of the convolution the concept of the kernel is different. In the case of pooling a kernel is just a window of of consideration from the input. The max element in the window is returned and the window is slide down the vector. Mathematically this is shown in equation 2.2

$$m(X) = max\{x_1, \ldots, x_n\} \tag{2.2}$$

2.1.3 Threshold Non-linearity

The Threshold non-linearity is a simple function defined as $f(x) = max(0, x)$. The idea behind this function is to reduce computation in the sense that the function creates a sparse output. The other motive for the threshold non-linearity is that it does not experience the

gradient vanishing problem as severely as other functions such as the Sigmod function due to the saturation tendencies of the Sigmod function.

2.1.4 Linear Layer

A linear layer is a simple linear transform of the input vector. This transform is defined by a weight matrix $W \in \mathbb{R}^{nxm}$ with an input vector $\mathbf{x} \in \mathbb{R}^m$ and output vector $\mathbf{y} \in \mathbb{R}^n$ defined by equation 2.3.

$$M\mathbf{x} = \mathbf{y} \tag{2.3}$$

2.1.5 Dropout

Dropout [**?**] is a form of regularization that assigns a probability to each input neuron of dropping the weights and connections of the neuron from the output. This omission of neurons forces the network to learn several different representations for each input. This forced representation robustness prevents features from co-adapting to each other which results in a representationally more powerful network. Dropout has shown to improve results for tasks as diverse as image recognition to drug discovery.

2.1.6 LogSoftmax

Finally the LogSoftmax [**?**] module is addressed. The LogSoftmax is a loss module which is used to predict a class from K independent classes. These features make LogSoftmax suitable for classification tasks. Mathematically the LogSoftmax function is defined for an individual element of the input in equation 2.4.

$$f(x)_i = \ln\left(\frac{e^{x_i}}{\Sigma_j e^{x_j}}\right) \tag{2.4}$$

2.2 Implementation Architecture

The model presented in this thesis is a deep Convolution Neural Network which is based off the network proposed in [**?**] for the task of character level Natural Language processing tasks such as Sentiment Analysis. After modifications the resulting network contains 7 convolution layers all of which are followed by a Threshold non-linearity. This particular non-linearity is similar to the Rectified Linear [**?**] (ReLu) activation function and is defined as $f(x) = max\{0, x\}$. All but the last convolution layer are followed by a max pooling layer. The details of the convolution layers such as kernel size can be found in table 2.1.

Table 2.1: Convolution Layers

Layer Number	Feature	Kernel Size	Pooling Size	Stride
1	256	15	3	1
2-6	256	3	3	1
7	128	3	N/A	1

At the top of this network are two fully connected layers each followed with another Threshold non-linearity and a Dropout layer with Dropout probability set to .5. The final Threshold is feed into a LogSoftmax layer. Feeding the non linearity directly into the LogSoftmax layer might seem like an incorrect thing to do but it in fact resulted in an accuracy gain of roughly 2 percent. Details for the linear layers are found in table 2.2.

Table 2.2: Details of Linear Layers

Layer Number	Output Number	Dropout Probability
8	1024	.5
9	11	.5

2.2.1 Data Set

Experiments are performed with the data set created in [**?**] to train their MultiLoc algorithm. The data set consists of 11 different sub-cellular protein locations and contains a total of

5959 unique sequences.

Table 2.3: Data Set Number for Training and Test size

Protein	Train Number	Test Number	Total
Chloroplast	359	90	449
Cytoplasmic	1128	283	1411
Endoplasmic reticulum	158	40	198
Extracellular	674	169	843
Golgi	120	30	150
Lysosomal	82	21	103
Mitochondrial	408	102	510
Nuclear	669	168	837
Peroxisomal	125	32	157
Plasma membrane	990	248	1238
Vacuolar	50	13	63
Total	4763	1196	5959

It can be seen from table 2.3 that this data set is heavily unbalanced with respect to the number of elements in each class. As an example of this unbalance; the smallest class, being the Vacuolar class, contains only 63 examples and the largest class, being the Cytoplasmic class, contains 1411 elements. This heavily unbalanced scenario is not favourable for training of Convolution Networks. Ideally each class would have roughly an equal amount of members and each mini batch would contain an equal amount of elements from each class during training. With such an unbalanced data set the ideal training scenario is not possible.

Another unfavourable aspect of this data set is the extreme variance in protein length. The shortest sequence is only 13 amino acids long while the longest is 5430 with an average amino acid sequence length of 500

For training and testing the data set was split into a 80% train and 20% test split. Such a split left 4763 data points for training and 1196 data points for testing.

2.3 Input Processing

2.3.1 Insights

One of the most important reasons for choosing the Neural Network from [**?**] is how the model handles inputs. Inputs are quantized at a character level where each character is mapped to a 1-of-k encoding where k is the number of characters recognized by the network. The set of the k characters is referred to as the alphabet. The alphabet of amino acids can be found in Table 1.1. In [**?**] the network dealt with an alphabet size of 70 characters which consisted of 26 English characters (only one case) 10 digits, the new line character and 33 other characters. Out of character elements where encoded as the all zero vector.

This character level alphabet, intuitively, is ideal for biological sequences. The reason for this is multifaceted. First the alphabet size is very small, in the Case of DNA/RNA the alphabet size is 4 while the alphabet size for amino acids is 20. Next many of the difficulties experienced in natural languages are not found in biological sequences. For example, biological sequences will not have out of alphabet characters. Overcoming uncertainty in the input is a great benefit for using this character level encoding in the domain of biological sequences. Lastly this simplifies the task of data cleaning. All the user needs to do is provide there biological sequences into a 1-of-k representation and then the interested user will be ready to feed the input into the model.

2.3.2 Details

For the experiment performed in this thesis the input sequences that are shorter than 3042 elements are extended with zero padding and longer sequences are naively truncated. In [**?**] the protein sequences were truncated to a length of 1000. When truncating, the middle of the protein was removed due to the fact that N- and C-terminal regions are known to contain sorting signals [**?**]. For this work the middle sequence truncation is viewed as expert knowledge. Since one of the main objective of this thesis is to reduce expert knowledge

needed to perform sub-cellular protein localization and attempt to omit all expert knowledge truncation is done on any amino acid exceeding the length of 3042 elements.

The input used in [**?**] was converted from a mapping between a character from the amino acid alphabet to a concatenated string which contains the 1-of-k encoding of the character, the column from the Blosum80 substitution matrix associated with the amino acid, the column from the HSDM substitution matrix associated with the amino acid and then the sequence profile of the amino acid. The information obtained from the substitution matrices and the sequence profile are viewed as hand engineered features in this thesis. This view coupled with the motivation to remove as much biological expert knowledge and hand engineering from the task is why a 1-of-k encoding is desired.

Chapter 3

EXPERIMENT SETUP AND RESULTS

3.1 Setup

Torch7 [**?**] is used to implement the model and Stochastic Gradient Descent (SGD) to train the model. The mini-batch size is set to 32 with an initial learning rate of .005. Learning rate reduction is performed once at the 15000th epoch to .0025 and Momentum is set to .9. A total of 20 eras are run, where an era is defined as 500 epochs. Input length is set to the fixed size of 3042.

It was observed that the mini-batch size made a large impact on the performance of the model. Changing to a smaller mini-batch, such as 16, lead to only achieving an accuracy of about 77 percent. Using a larger mini-batch also affected the model accuracy in a similar way. These facts are stated here because, apparently, the mini-batch hyper parameter heavily impacts the performance of the model and is worth being brought to the attention of the interested reader.

A 5-fold evaluation is performed. Due to the small data size each fold overlaps each other and is not mutually exclusive. For each fold different methods were used to separate each class into an 80/20 train/test split with each method extracting non-identical folds but at

the same time non-unique folds. Care was taken to make sure that for each fold the same number of elements could be found in the training and testing data sets. This means that the training data size and the testing data size was always the same for each fold. Results focus on the best fold but the average of all 5 folds is also presented in the next section as well as the standard deviation of the average results.

3.2 Results

The model was able to achieve a best test accuracy of 83.6%. On average the model was able to achieve an accuracy of $79.78\% \pm 4.21$. A comparison between the overall accuracy of the best single model and other single model solutions can be found in table 3.1.

Table 3.1: Model Comparisons

Model	Accuracy
Convolution	83.6%
R-LSTM	87.9%
A-LSTM	85.4%
MulitLoc	76.7%

Analysis of individual classification accuracy for each class can be found in Table 3.2. Due to the large imbalance of the training data the assumption was made that the model would perform poorly on the classes with a small number of training examples and better with classes that have larger number of training examples. For the most part the assumption is correct. There are however a few interesting exceptions to the assumption. For example, the Gogli class with only 120 training examples obtains an 80.% training accuracy in contrast to the Peroxisomal class which achieves a 28.3% accuracy with a similar training example size of 125. On the other end of the spectrum the class with the largest number of training examples, the Cytoplasmic class, only achieves an 87.2% accuracy while the Plasma membrane class achieves 98.7% accuracy with 138 fewer training examples. Another anomaly is the Chloroplast class which achieves a 90.6% accuracy with only 359 training examples.

Table 3.2: Best Fold Classification Accuracy per Class

Protein/Class	Test Accuracy	Train Size
Vacuolar	23.2%	50
Lysosomal	33.5%	82
Golgi	80.6%	120
Peroxisomal	28.3%	125
ER	47.8%	158
Chloroplast	90.6%	359
Mitochondrial	79.0%	408
Nuclear	71.4%	669
Extracellular	96.5%	674
Plasma membrane	98.7%	990
Cytoplasmic	87.2%	1128

To evaluate the slightly counter intuitive classification behaviour stated above an analyse of the confusion matrix from the thesis models testing experiments with results can be found in table 3.3. A reasonable hypothesis to the testing results is that the classes with very few training examples will classify to random classes at test time. One can see that this is not the case when focusing on the Vacuolar, Lysosomal and Peroxisomal classes. The confusion matrix shows that in all three of these classes the low test accuracy does not manifest as a mapping to random classes but a struggle between two or three classes. The confusion matrix shows that the Vacuolar class is unable to decide between classification to the Vacuolar and Extracellular class. The Lysosomal class also has trouble differentiating between the Extracelluar class as well as the Plasma membrane class. The Peroxisomal class struggles between itself and the Cytoplasmic class in the majority of examples.

Other unexpected accuracy scores seem to fall into the same idiom. For example the Cytoplasmic class tends to incorrectly classify to the Nuclear class while the Nuclear class tends to classify as the Cytoplasmic class for the majority of the errors experienced for the either class.

It is encouraging to see that even with very small training examples the model is able to learn something useful and classify classes with smaller training examples in a similar manor to other classes with more training examples. For future work a deeper network might be

Table 3.3: Confusion matrix from the best model from this paper.

Confusion Matrix											
ER	20	2	11	0	6	0	0	1	0	0	0
Golgi	1	24	2	0	1	1	0	1	0	0	0
Extracellular	5	1	163	0	0	0	0	0	0	0	0
Lysosomal	2	0	7	8	4	0	0	0	0	0	0
Plasma membrane	0	0	4	0	244	0	0	0	0	0	0
Vacuolar	0	2	7	0	0	4	0	0	0	0	0
Chloroplast	0	0	0	0	0	0	82	3	5	0	0
Mitochondrial	0	0	2	0	1	0	7	80	12	0	0
Cytoplasmic	0	0	2	0	2	0	6	2	246	24	2
Nuclear	0	0	2	0	3	0	2	2	39	120	0
Peroxisomal	0	1	0	0	3	0	0	0	17	1	10

able to correct the errors seen in these experiments because it should be able to evaluate the classes at a higher level of abstraction.

In table 3.4 the classification accuracy of each class for the average of the five folds is shown as well as the standard deviation. While the spread is large for the majority of the examples one class stands out and that is the Mitochondrial class. The reason the standard deviation is so high for the Mitochondrial class is that one of the five folds was unable to correctly classify any of the test samples as Mitochondrial but instead classified the majority of the examples as Plasma membrane. The other 4 folds all classified Mitochondrial proteins with around 80% accuracy. Why only one class in one fold did so poorly is something that needs to be investigated.

For the most part the classification behaviours seen in the best fold were still seen in the average of the 5 folds. The only difference seen between the best fold and the average of all folds is that the classification accuracy went down and, as stated before, the anomaly of one fold not being able to classify the mitochondrial class correctly even one time.

Table 3.4: Average Fold Classification Accuracy per Class

Protein/Class	Test Accuracy	Train Size
Vacuolar	$13.3\% \pm 5.07$	50
Lysosomal	$36.0\% \pm .983$	82
Golgi	$66.6\% \pm 5.36$	120
Peroxisomal	$14.8\% \pm 5.71$	125
ER	$49.2\% \pm 2.36$	158
Chloroplast	$86.9\% \pm 3.23$	359
Mitochondrial	$65.6\% \pm 37.55$	408
Nuclear	$68.8\% \pm 4.15$	669
Extracellular	$91.4\% \pm 5.49$	674
Plasma membrane	$97.3\% \pm 1.72$	990
Cytoplasmic	$85.4\% \pm 1.64$	1128

3.3 Comparisons

Next comparisons of the model in this thesis are made against the R-LSTM from [**?**]. While the R-LSTM achieves a better test accuracy of 87.9% compared to the thesis model 83.6% the kinds of errors that each model makes are evaluted next. The hope is that each model makes different errors and would be able to complement each other well.

Starting with the Vacuolar class one can see from the confusion matrices of the two models, with the confusion matrix of R-LSTM being found in table 3.5, that both models tend to make the same errors of classifying the Vacuolar Class to the Extracellular class. Next, with focus on the Lysosomal class for each model, it is apparent that both models tend to incorrectly classify Lysosomal as Extracellular. The Golgi class has a high level of test classification accuracy for both models. Peroxisomal gets misclassified frequently as the Cytoplasmic class in both models. Finally Cytoplasmic is misclassified as Nuclear in each model while Nuclear is misclassified as Cytoplasmic in each model.

From examining the confusion matrix of each model it is apparent that both models have the same difficulties differentiating between the same classes. While the R-LSTM has a higher classification accuracy it also incorporated extra information to each amino acid beyond the 1-of-k mapping as well as used domain knowledge with respect to how to truncate the input

Table 3.5: Confusion matrix of the R-LSTM Model

Confusion Matrix											
ER	26	1	0	0	8	1	0	0	0	3	0
Golgi	1	28	0	0	0	0	0	0	0	1	0
Chloroplast	0	0	82	3	0	0	5	0	0	0	0
Cytoplasmic	0	0	1	266	0	0	3	12	0	0	0
Extracellular	0	0	0	1	166	0	0	0	0	1	0
Lysosomal	0	0	0	0	5	12	0	0	0	3	0
Mitochondrial	0	0	2	5	0	0	94	1	0	0	0
Nuclear	0	0	0	27	1	0	3	137	0	0	0
Peroxisomal	0	1	0	10	0	0	0	1	18	2	0
Plasma membrane	0	0	0	0	5	0	1	1	0	241	0
Vacuolar	0	0	0	0	7	0	0	0	0	1	5

amino acid sequences and a different architecture. Showing that convolution network can learn the same features as an LSTM is noteworthy because of the open question with respect to how different the two architectures are in what they learn.

3.4 Experimental Discussion

For this thesis three models where developed and the the best performing model was choose. The training behaviour of these three models is worth discussion. It was observed that when a more standard model, which omits the threshold and Dropout between the last linear layer and the LogSoftmax, was used the model was able to quickly achieve 100% training accuracy on the training set. Unfortunately the model was only able to achieve around 80% test accuracy and was unable to continue learning due to its perfect testing classification.

By mistake a threshold non-linearity and Dropout Layer was added between the final linear layer and LogSoftmax layer. The resulting network was no longer able to achieve 100% classification accuracy on the training data but instead hovered around 2%. The inability for the network to achieve 100% training accuracy allowed the network to continue to learn and eventually achieved 83.6% classification accuracy on the test data. Ultimately it is not possible to extract any more accuracy out of the network in this work by adding

additional layer for the reason that the feature size coming out of the final max pooling layer is of size 1 and thus new models needs to be investigated.

The over fitting behaviour of the three models are analysed in the next section. The training Accuracy of all three models can be found in appendix 5.

3.5 Over Fitting Behaviour

In this section the over fitting behaviour of initial three models that where experimented with for this thesis is discussed. These models are identical except for the operations performed in-between the last linear layer and the LogSoftmax layer. The models and how they differ can be found in table 3.6.

Table 3.6: Over Fitting Model differences

Model Name	Module Between Last Linear Layer and LogSoftmax
Model 1	None
Model 2	Dropout
Model 3	Threshold and Dropout

For simplicity each model is termed model 1, model 2 and model 3 with the models being described as follows. Model 1 has no operations between the last linear layer and the LogSoftmax (worst case of overfit). Model 2 has a Dropout Layer between the last linear layer and the LogSoftmax (second best case of overfit) and finally model 3 has a Dropout and Threshold non-linearity (best reduction of overfit and best overall model) between the last linear layer and the LogSoftmax.

To evaluate the over fitting behaviour the graphs from the objective loss of each model is analysed over roughly 20 eras with each era consisting of 500 epochs. Figure 3.1 shows the objective loss in model 1 quickly begins to diverge after reaching era 8 (epoch 4000). The divergence seen in the figure shows a classical example of over fitting. While model 1 does have a Dropout layer between the first and second linear layer it did not provide enough

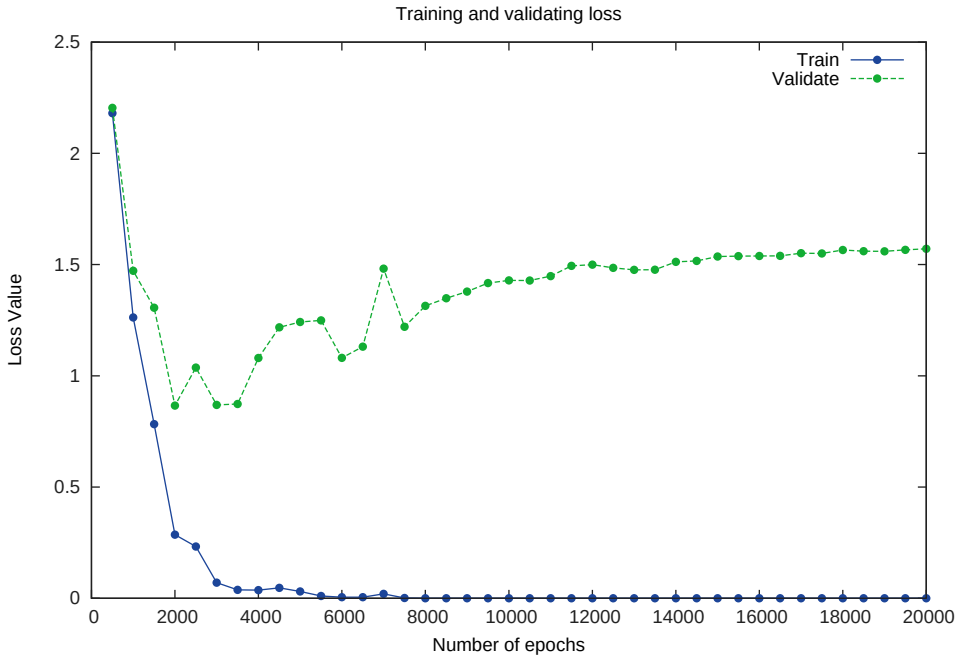


Figure 3.1: Objective Loss of Model 1 without Dropout and Threshold

regularization to prevent the model from over fitting on the training data.

Figure 3.2 shows that model 2 is able to delay the affects of over fitting until the training loss approached zero around era 14 (epoch 7000). Once the training loss of model 2 approached zero the test objective loss began to increase showing the model over fitting.

Finally the behaviour of model 3 with Dropout and a threshold non-linearity can be seen in figure 3.3. Noting that model 3 is the best performing model from experiments it can be seen that the model seem to not experience over fitting in the testing objective loss. Once the training objective for this model approached zero the test object stays steady and smooth while model 2 with only Dropout between the last linear layer and LogSoftmax began to diverge. Of course the fact that the training objective value approaches zero means that the ability for the models to continue to learn and improve the testing classification accuracy is severely diminished the lack of over fitting is a good signifier of a model that generalizes better than the other two models.

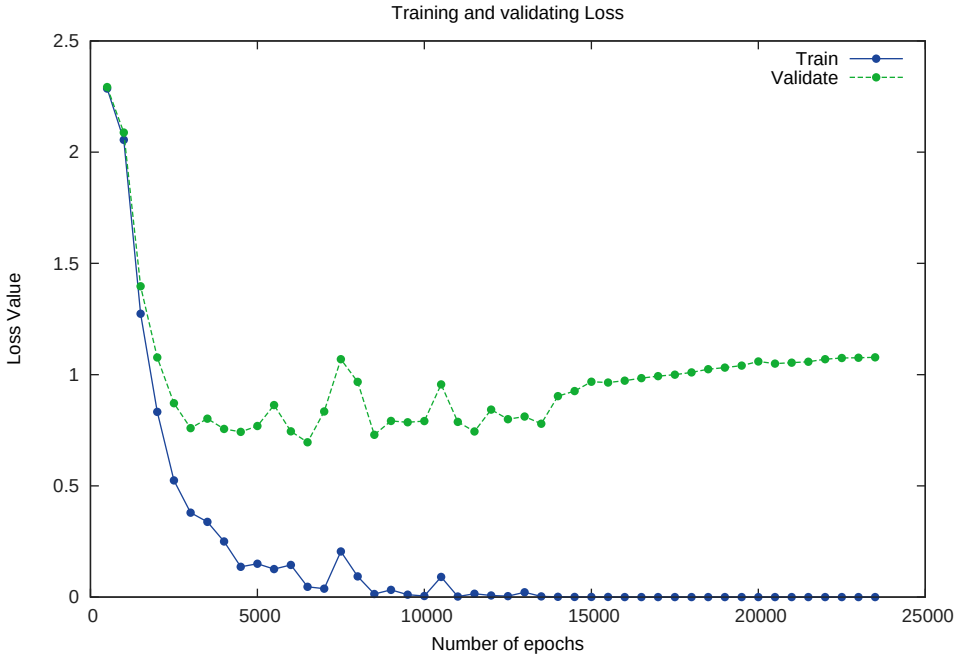


Figure 3.2: Objective Loss of Model 2 with just Dropout.

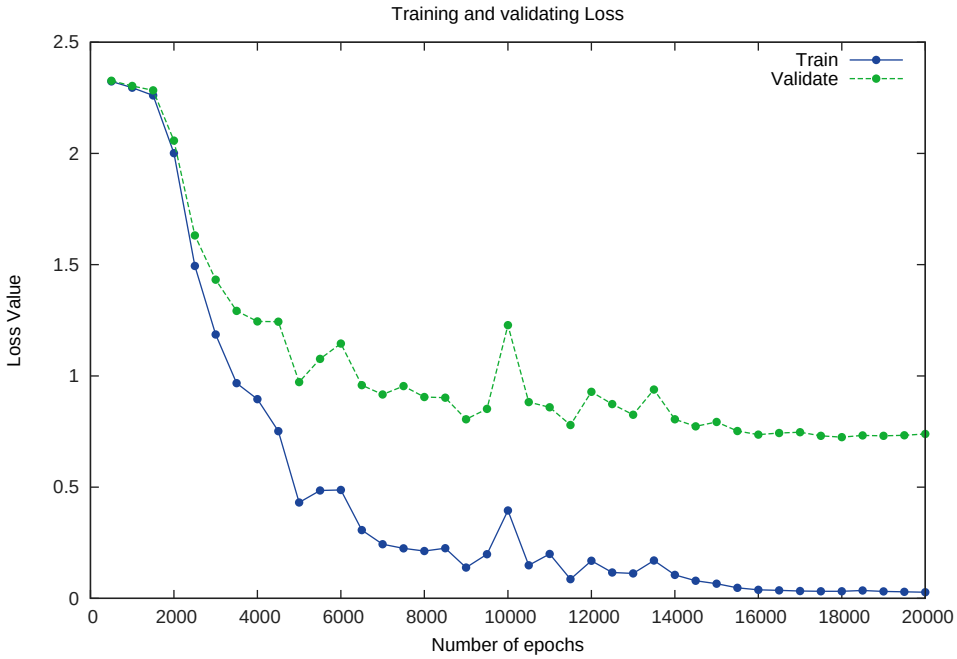


Figure 3.3: Objective Loss of Model 3 with Dropout and Threshold

Chapter 4

CURRENT PROBLEMS AND FUTURE WORK

4.1 Current Problems

Some existing problems that were experienced are briefly discussed below. This brief summary rounds out the most egregious errors that should be looked into for the purposes of future work and further progress on the task of sub-cellular protein localization in the scope of Natural Language Processing.

4.1.1 Experimental Shortcomings

Experiments On More Data Sets

Lack of experimentation on RNA and DNA as well as other proteins datasets are glaringly absent from this work. It might be the case that experimentation on other data sets show that the method contained in this thesis performs poorly on other types of biological sequences. It also might be the case that the model performs poorly on other data sets of proteins for the task of Sub-cellular protein localization. More experimentation on other sets is needed. Until more experimentation is performed on other data sets it is not possible to determine how the method contained in this thesis performs or if the numbers obtained in this thesis

were just an artifact of the data set of experimentation itself.

Evaluation Metrics

The metric of analysis in this thesis was taken by the accuracy of classification overall as well as for each individual class. Some possibly interesting characteristics between how each class is classified or misclassified where observed but other metrics might be more informative. Some standard metrics in the Natural Language Processing community such as F-score, precision, recall as well as other metrics could bring more insight into the effectiveness of the method evaluated in this thesis.

Input Truncation

Possibly the most important current problem is the necessity to truncate the input to an arbitrary fixed size. This has two negative affects. The first problem arises in the added computational cost incurred by extending shorter sequences, keeping in mind the average sequence is only 500, to more than 3000. This scenario means computation time is lost because the first convolution layer has to slide across all 3042 characters even though the majority of them or all zeros.

The next problem incurred when truncating is the loss of information from sequences greater than the truncation length. By Truncating sequences that are longer than 3042 the model could be missing out on learning important features that might help it differentiate between the classes that the model has the most trouble classifying.

4.1.2 Model Weaknesses

Deeper Networks

While the network focused on during this work was of considerable depth it is possible that a deeper network would be able to learn to differentiate classes better than the network discussed in this thesis. As can seen by the Objective function analysis in section 3.5 the

network has reached its capacity with respect to learning. If accuracy is to be increased while domain knowledge and hand engineering is to be kept low it is necessary that another architecture of more depth needs to be investigated.

Optimization Alternatives

The Learning method used, Stochastic Gradient Decent, might also be a problem in the experiments. Stochastic Gradient Decent is more fitting for learning tasks on very large amounts of data. The exact opposite situation is true in this work with a test set size of less than 5000 data points. An alternative learning algorithm might have given better results as well as taken less time to train.

4.2 Future Work

This section contains a set of data points that fit into the category of things that would have been nice to have known six months ago. The ideas in this section are also associated with potential problem that they would alleviate in this current work.

4.2.1 Residual Networks

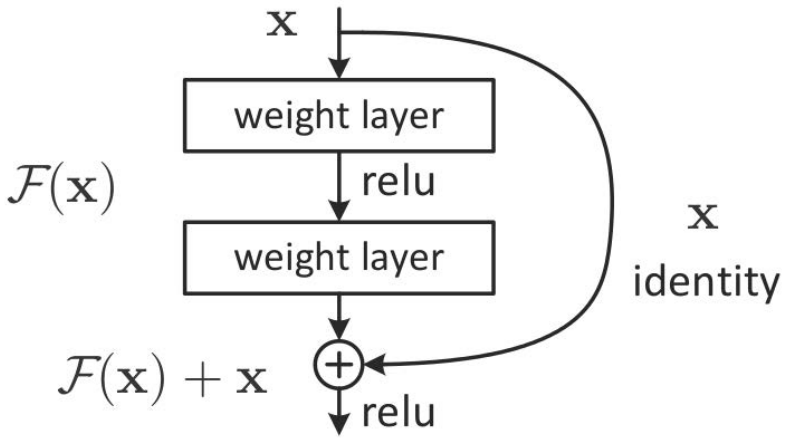
The most import future avenue to investigate is a new convolution neural network architecture developed by Microsoft referred to as a Residual Neural Network [**?**]. The Residual architecture has the potential to alleviate many of the short comings experienced in the network discussed in this work.

Depth

The first advantage that Residual networks posses is that they consist of many more convolution layers and as a result are much deeper than the more conventional convolution networks. The way that these networks are able to reach such impressive depths is because of two important facts.

The first factor is that pooling layers are no longer used save for a single pooling layer at the top of the network. The lack of pooling means that the learned features from each layer do not reduce in size like the networks which deploy many pooling layers. Combine the lack of pooling with convolution kernels of a very small size and the result is that learned features shrink in size slowly and many layers may be incorporated affectively.

The second factor is the use of skip connections in Residual Networks. Skip connections are connections from outputs of a certain layer and are then combined in some way to inputs of another layer that is not the immediate layer proceeding the output layer. A graphical example of a skip connect can be found in image 4.1. The benefit of using skip connects in very deep convolution neural networks is that training becomes easier. Several explanation as to why skip connects ease the training of Residual networks range from the ability to reduce the vanishing gradient problem to the ability of each layer to learn a deviation from the identity layer provided by the skip connect output.



[?]

Figure 4.1: Skip Connection Example in Residual Network.

A problem with using Residual networks for the task of pre-cellular protein localization is that they have been designed from image recognition problems and are therefore spatial architectures. Since Natural Language Processing problems are temporal in nature and image recognition problems are spatial it would require the development of temporal Residual

networks to tackle the problem of sub-cellular protein localization as a Natural Language Problem. Given the current software tools such as Torch7 creating such a network might not be an extremely difficult task and is undoubtedly worth looking into.

Variable Size Inputs

Residual networks are also able to handle variable size inputs without the need to normalize the inputs to some fixed size. Right away the problem of data padding and truncation is alleviated by the new Residual Network Architecture. However a new problem does arise in the scenario of variable size input. While the network can take a variable size input it will also output a variable size output. As an example, a Residual network that has 101 layers will take in an NXM matrix and output a tensor of size $512xIxJ$. Since I and J are variable there needs to be some method to standardize the size so it can take as input into some other layer such as a LogSoftmax Layer. Luckily there are pooling methods termed pyramid pooling [**?**] which take in a variable size input and are able to output a fixed size structure. Again, Currently pyramid pooling can only be found for spatial problems such as image recognition and would require research into the development of a temporal pyramid pooling method.

Residual Networks Potential

The combination of a very deep convolution architecture with the ability to handle variable size inputs while utilizing all information provided in each input has the possibility of improving the results of the task focused on in this paper. Residual networks could improve the task of sub-cellular protein localization by being able to utilize all information contained in all amino acids sequences at a reduced computational cost while being able to learn a more abstract representation of the protein sequences which would in theory reduce the problem of protein class confusion seen in this work as well as [**?**].

4.2.2 Alternative Regularization

If Residual networks do not turn out to be a promising improvement over the model discussed in this thesis then there is another idea that could be evaluated on the convolution network from this work.

Dropout Variation

A variation of Dropout regularization worth looking into is termed Annealed Dropout [**?**]. What annealed drop out is is a process of setting the Dropout probability to some high value at the beginning of training. While training evolves the Dropout probability gradually decreases until it reaches the value of zero. It is shown that Annealed Dropout works well for small data sets and outperforms regular Dropout as well as several other methods that focus on using Neural Networks on problems with small data sizes.

4.2.3 The State of Deep Learning

The reality is that the field of Deep learning is moving at an extremely rapid pace. Many problems that seem ill fitted to a Neural solution could quickly become a prime candidate due to unforeseen advances. It is because of this fact that may of the shortcomings of the methods in this thesis are addressed and also some possible solutions to those short comings are discussed. If experiments had been started three months later then residual networks might have been used on the problem addressed in this thesis instead of the network presently discussed. Annealed Drop out might have been investigated in this thesis if a discussion with a co-worker about the work in this thesis was had at an earlier date. All of the ideas in this section are worth investigated and are well enough defined that the difficulty to implement a new solution based of the ideas in future work would not be too difficult.

Chapter 5

CONCLUSION

In this thesis the task of Sub-cellular Localization of Proteins has been viewed as a Natural Language Processing task. It was shown that the task can be performed with zero domain knowledge by modifying an existing Character Level Convolution neural model. Comparing these results against other single model methods, the model in this thesis is able to achieve a higher test accuracy than an existing model that requires a large amount of hand engineering and biological knowledge. It is also shown that the NLP method makes similar classification errors as another Deep Neural model. This neural model outperforms the model in this thesis with respect to testing accuracy but also requires some Bioinformatics knowledge to transform the amino acid sequences into a model friendly input, while the model in this thesis only requires a 1-of-k mapping for each amino acid.

It is not possible to determine if the method contained in this thesis performs the task of Sub-cellular protein localization well due to the lack of experimentation on other data sets. All results obtained in this thesis could be an artifact of the data set that was experimented on and not a reflection of the effectiveness of the method evaluated on in this paper. More experimentation on other data sets as well as the evaluation of other metrics must be performed before it will be possible to determine if treating proteins as natural language for the task of Sub-cellular protein localization is an effective method. Also experimentation on other biological sequences such as RNA and DNA must be performed before it can be

determined that treating all biological sequences purely as natural language is beneficial in anyway.

BIBLIOGRAPHY

Bibliography

[1] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

[2] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3):13:1–13:37, June 2008.

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

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Annette Höglund, Pierre Dönnes, Torsten Blum, Hans-Werner Adolph, and Oliver Kohlbacher. Multiloc: prediction of protein subcellular localization using n-terminal targeting sequences, sequence motifs and amino acid composition. *Bioinformatics*, 22(10):1158–1165, 2006.

[6] Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen, and Ole Winther. Convolutional lstm networks for subcellular localization of proteins. In *Algorithms for computational biology*, pages 68–80. Springer, 2015.

[7] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[8] Andreas Prlić, Francisco S Domingues, and Manfred J Sippl. Structure-derived substitution matrices for alignment of distantly related sequences. *Protein Engineering*, 13(8):545–550, 2000.

[9] Dimitri Palaz, Ronan Collobert, et al. Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of Interspeech*, number EPFL-CONF-210029, 2015.

[10] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer, 2010.

[11] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[13] Fei Sha and Lawrence K Saul. Large margin hidden markov models for automatic speech recognition. In *Advances in neural information processing systems*, pages 1249–1256, 2006.

[14] Olof Emanuelsson, Søren Brunak, Gunnar von Heijne, and Henrik Nielsen. Locating proteins in the cell using targetp, signalp and related tools. *Nature protocols*, 2(4):953–971, 2007.

[15] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(9):1904–1916, 2015.

[18] Steven J Rennie, Vaibhava Goel, and Samuel Thomas. Annealed dropout training of deep networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 159–164. IEEE, 2014.

APPENDIX

APPENDIX: Behavior of Three Models

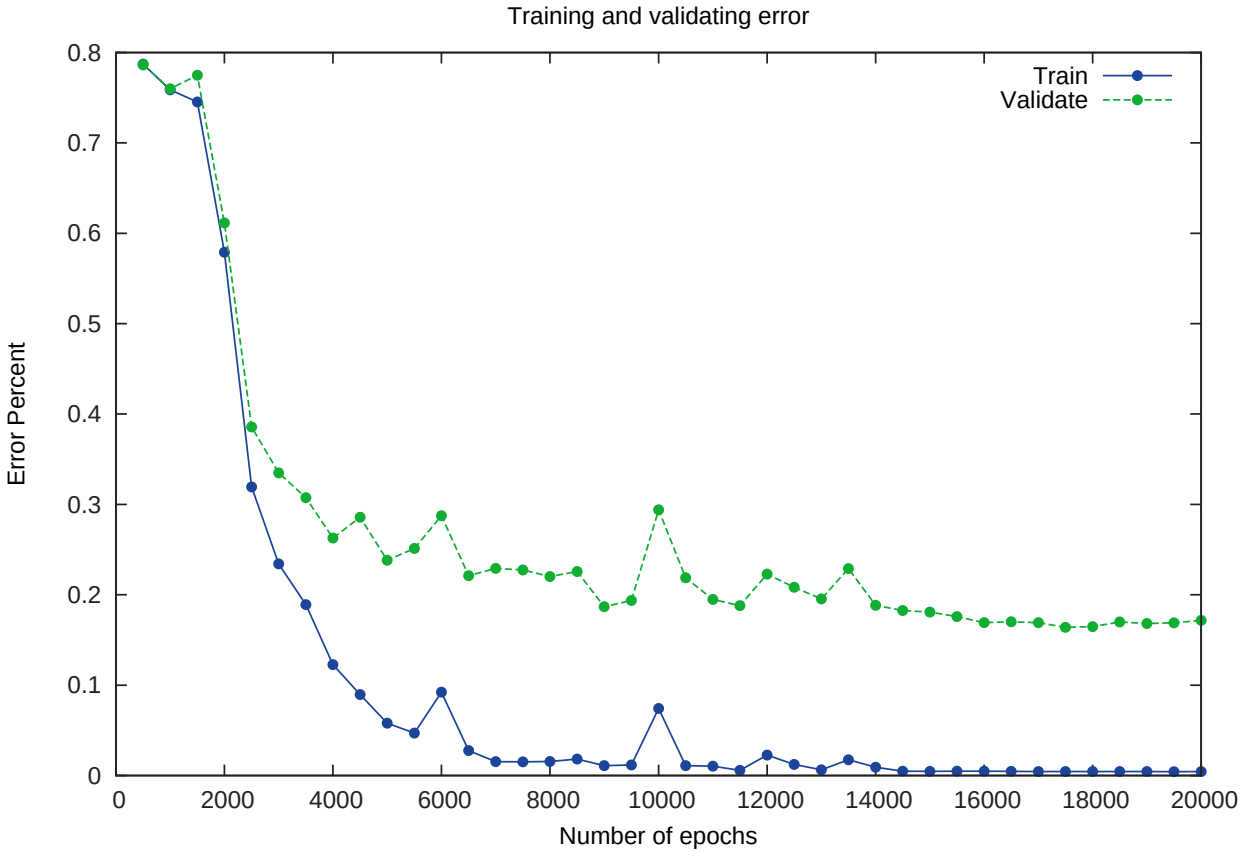


Figure 1: Error value of Model 1 (best model)

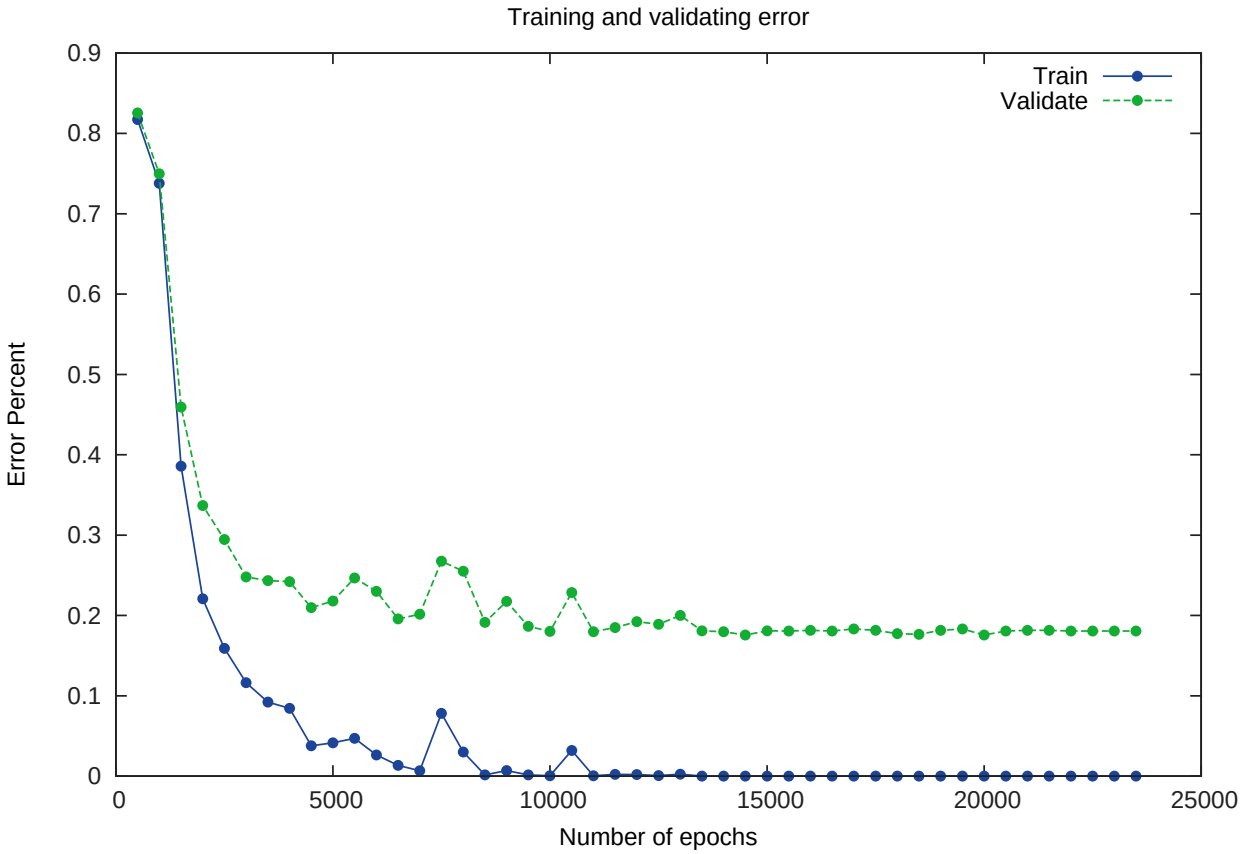


Figure 2: Error value of Model 2 (with just Dropout)

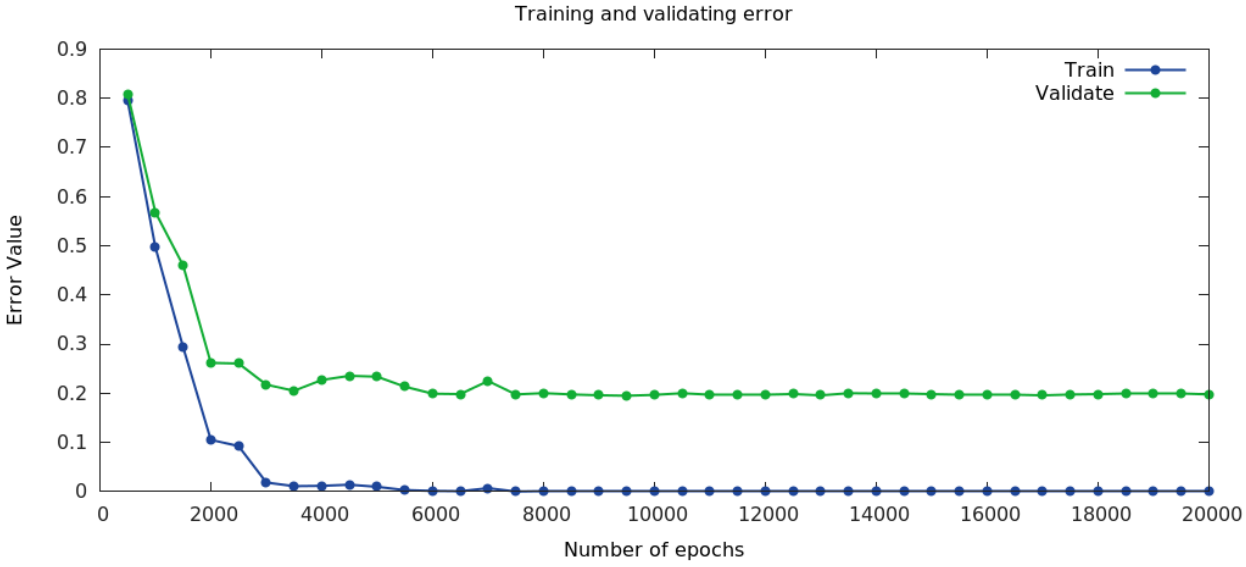


Figure 3: Error value of Model 3 (no Dropout or Threshold)