30 09 2019

|  |
| --- |
| Machine learning  Signal peptide prediction with Sec2Vec embeddings  Thomas Heinzinger \*  \*To whom correspondence should be addressed.  Associate Editor: Michael Bernhofer  Abstract  **Motivation:** The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.  **Results:** The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.  **Availability:** <https://github.com/tomthun/Masterpraktikum>  **Contact:** [heinzinger.thomas@gmail.com](mailto:heinzinger.thomas@gmail.com)  **Supplementary information:** Supplementary data and information are available on request. |

# Introduction

Protein allocation within and outside of cells are important and complex biological tasks, often guided specific signal peptides (SP) contained in the N-terminus of protein sequences. SPs are usually around 16-30 amino acids long and used by every organism, including Archaea, Eukarya and Bacteria. Regarding the general secretory pathway, the organisms use different approaches whereby protein translocation in prokaryotes is directed across the plasma membrane and the endoplasmic reticulum membrane in eukaryotes. Therefore, proteins endowed with a signal peptide are resident in the endoplasmic reticulum and Golgi apparatus, secreted proteins and proteins inserted in plasma membranes. In sum, protein destination and function are intertwined with their according signal peptides, which is why it is key to find reliable methods to predict these. Another main parameter to predict is the position at where the signal protein is cleaved off from its host protein, which is also known as the cleavage site. This happens during or after membrane translocation by digestion through signal peptidases. Over the last decade evolutionary information (e.g. evolutionary couplings1) have become most popular and efficient as fundamental data for prediction tools. Studying a protein over time yields valuable insights on how e.g. mutations (insertions, deletion, etc.) have effects on protein properties (BLOSUM2).

One of the first publicly available methods for signal peptide prediction is SignalP3. It has been continuously improved over the last decade: Frist, simple artificial neural networks were used for prediction3, in the later versions hidden Markov models4 and more complex deep learning architectures5,6 were applied, resulting in improved cleavage site predictions and discrimination of signal peptides and transmembrane helices.

In its latest revision, the so called SignalP5.06, the authors also provided a publicly available dataset over 20758 proteins. The proteins in the dataset are annotated in a 3-line FASTA format, where three distinct signal peptides are being distinguished:

* Sec/SPI: "standard" secretory signal peptides transported by the Sec translocon and cleaved by Signal Peptidase I (Lep) and are exclusive to eukaryotes
* Sec/SPII: lipoprotein signal peptides transported by the Sec translocon and cleaved by Signal Peptidase II (Lsp)
* Tat/SPI: Tat signal peptides transported by the Tat translocon and cleaved by Signal Peptidase I (Lep)

Here, we utilized the dataset to predict these signal peptides with another new deep-learning method for protein properties called SecVec7. Originally used for protein folding and structure prediction its architecture allows for overarching usage regarding proteins. So-called long short-term memory networks8 (LSTMs) process the information of protein properties into continuous vectors (embeddings). First, this idea arose in the field of natural language processing9 since syntax and semantics of language can be learned by the probability distribution of words in sentences. Depended on the context of a sentence, words are then differently parameterized. This is advantageous since two identical words with possible different meanings can be contextually distinguished based on the composition of the sentence in which they are used. LSTMs can be similarly applied to proteins whereby proteins are seen as “sentences“ and amino acids as “words”. The models are fed with databases of proteins (e.g. Uniref50) in order to learn different sentences/protein compositions. Thereby no evolutionary information is used implicitly or explicitly.

A notable improvement is the speed at which embeddings can be created. Once a LSTM model is fully trained (note that this consumes most of the time) creating embeddings takes about 0.03 seconds7. Compared to that, most commonly methods are built around evolutionary information and couplings10,11 by alignment of similar proteins. However, those algorithms are becoming increasingly computationally costly since the number of UniProt entries grow faster every year through next generation sequencing12 methods. Even fast and highly optimized algorithms such as the HHblits313 need several minutes for finding and aligning similar proteins. Furthermore, evolutionary information is still missing for proteins in UniProt, e.g. the entire “Dark Proteome”14 which consist of less-well studied proteins although they are important for function15.

As part of this work we transformed the FASTA-dataset provided by SignalP5.0 into its vector representation using SecVec. Then we used the resulting embeddings and trained a simple one-layer convolutional neural network (CNN) with the addition of a conditional random field. Next, to assess the predictive power of the embeddings we distinguished between two levels: per-residue (word-level) and per-protein (sentence-level). Regarding per-residue level, we predicted three different signal peptides equally as in the original publication of SignalP5.0. Non-signal peptides can be differentiated between inner, outer and trans-membrane, but are merged into ‘*Others*’. On per-protein level, we simply observed if the per-residue prediction contains a signal peptide and if so, label the protein with the according SP type. Finally, to benchmark the efficiency of our method we used an according benchmark-dataset provided by SignalP5.0. This dataset a subset of proteins of the complete dataset whereby included ones are proteins that are less likely to be similar to others. Hence, if such can be detected with high likelihood the integrity of the network and comparison is ensured.

The results show that it is possible to use SecVec embeddings to predict distinct signal peptides down to residue level. In this project we reached an overall Matthews correlation coefficient (MCC) of around 0.868 on the benchmark. Compared to that SignalP5.0 reached around 0.94. Although the latter achieved better results it is notable that here most simplistic architectures were utilized. Asides, it was possible to show that an abstract contextualization of proteins into vectors might inherit more information than a combination of protein properties and evolutionary information. The project successfully demonstrates the prediction of signal peptides and that SecVec embeddings are not only applicable for learning protein structures.

# Methods

**Data**: We trained and benchmarked a simple convolutional neural network on embeddings based on the FASTA files provided by the SignalP5.0 website. The datasets are publicly available under:

<http://www.cbs.dtu.dk/services/SignalP/data.php>

The embeddings have been created by applying the SecVec method. A detailed tutorial on how to use the method is given on the following GitHub page:

<https://github.com/Rostlab/SeqVec>

As stated in the methods section of SignalP5.0 the UniProt Knowledgebase release 2018\_0416 has been for data gathering. Further preprocessing of the data can be found in the publication (ref6). The training and benchmark datasets contain all 20 standard residue letters, 20758 and 8809 proteins respectively and every protein as well as contained residues are distinguished between three distinct signal peptides;

* Sec/SPI: "standard" secretory signal peptides transported by the Sec translocon and cleaved by Signal Peptidase I (Lep) with residue annotation **S**
* Sec/SPII: lipoprotein signal peptides transported by the Sec translocon and cleaved by Signal Peptidase II (Lsp) with residue annotation **L**
* Tat/SPI: Tat signal peptides transported by the Tat translocon and cleaved by Signal Peptidase I (Lep) with residue annotation **T**

as well as non-signal peptide annotations. Since non-signal peptide residues can be separately located in cytoplasm, transmembrane and extracellular, they are annotated with **I**, **M** and **O**. The header further contains information about animal kingdom (eukaryote, archaea, gram – positive/ negative), Uniprot identifiers and separation into specific protein subsets / “splits”. The splits are relevant for cross-validation and observation of the learning progress of the CNN. Furthermore, one split will be excluded at the beginning of the training which is later extracted from the benchmark dataset and used for comparisons. The benchmark dataset has the same design layout as the normal dataset with the difference that included splits only contain proteins that are relatively unique in their properties. That enables an unbiased comparison between the approaches of SeqVec and SignalP5.0.

At the current revision SeqVec produces an output vector of 1024\*(protein length) of floating numbers that can range from negative to positive. The major reason for that is the network architecture of the algorithm which allows to capture multiple probable protein properties and report them in such a manner.

Data processing: Both datasets are processed in phyton version 3.7.3 (<http://www.python.org>) and learned/ evaluated with the help of the deep-learning framework pytorch17. Since proteins are generally around 466 and signal peptides around 23 residues long, both datasets inherently have class imbalance between non signal and signal peptide residues. To strike this problem, first, proteins and embeddings have been shortened to a length of 70. In theory this should not lead to a loss of training information as SecVec was applied on the whole protein before. The information of residues at the far end of a protein is already be implicitly contained in the ones at the beginning. This indeed reduced class imbalance although data analysis revealed that there is still a strong disequilibrium as seen as in table:

**Table 1.**Relative distribution of signal peptide residue annotations across training and benchmark datasets.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Signal | Training dataset | Benchmark Dataset |
| Non | I | 0.70 | 0.83 |
| SP: | M | 0.03 | 0.02 |
|  | O | 0.18 | 0.10 |
| Sp: | S | 0.05 | 0.02 |
|  | T | 0.01 | 0.01 |
|  | L | 0.02 | 0.02 |

A similar imbalance can be observed at a protein level where 0.74 for training and 0.86 percent for benchmark do not have a signal peptide. Secondly, we decided to address this disparity by weighting each class individually (class weighting is enabled in pytorch). Moreover, 187 proteins that are shorter than 70 have been identified. Embeddings and labels of those have been padded accordingly (with no effect on learning) until they have the desired length. Equally to SignalP5.0 non-signal proteins have been summarized to one class before learning starts. In the following it will only be distinguished between non-SP and the three different kinds of SP.

Model architecture: For benchmarking purposes, we used the similar layout as described in SignalP5.0:

1. 2D convolution with 1024 input units, 64 filters, kernel widths (5,1) and padding of (3,0).
2. 2D convolution with four filters (matching the number of classes) and a kernel width of one. No padding is needed in this specific case.
3. Conditional random field (crf) for predictions. The forward–backward algorithm was used to calculate each individual and marginal residue label probability of the at the specific position in its protein. The global most likely label assignment for the entire sequence was done via Viterbi decoding.

SignalP5.0 also uses a specific bidirectional LSTM to enrich data as well as include taxonomic group information in the model. Since data enrichment has already been done by SecVec we skipped this and previous layers. Also different is the usage 2 dimensional convolutional networks in order to capture different residue properties (recall that SecVec creates a vector of 1024 for each residue). Therefore, specific padding (3,0) is needed. To avoid that overfitting18, dropout19 as well as batch normalization20 layers have been added. The loss function is calculated through the negative log likelihood between true and predicted labels returned by the forward function of the conditional random field implementation. Further reading on that topic and the used programming suite is available on GitHub:

<https://github.com/kmkurn/pytorch-crf>

All parameters were optimized using Adam optimization tool21 with mini batches of size 128 and learning rate 0.01.

The previously described techniques improve the performance on both the signal type and the cleavage site prediction though it is noteworthy how the addition of a conditional random field improves the results: Compared to a model without, it enhances both predictions types on a global (protein) as well as residual level. We also observed that the total number of signal peptide gaps and mixed typed predictions are drastically reduced. Gaps in signal peptide predictions and mixtures of the three types are not biologically meaningful and should not happen (ref5). Still even after application of the crf a few of such problems remain. In both cases simple postprocessing helps: First gaps are filled based on the surrounding signal peptides. Secondly, if a mixed type prediction is present, lower frequent types are replaced by the most abundant one. Table 2 shows a comparison between the performance of the model using a crf and a model without one after postprocessing.

**Table 2.** Comparison between the performance of a model with and without usage of a crf. Δ CS = mean residue deviation from the actual cleavage site. Both models have been trained with the same hyperparameters.

|  |  |  |  |
| --- | --- | --- | --- |
|  | WIth Crf |  | Without crf |
| Δ CS | 0.543 |  | 1.095 |
| residual MCC | 0.894 |  | 0.824 |
| Global Mcc | 0.904 |  | 0.813 |

# Results

## Data Structure This is Heading 2 style this is heading 2 style

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

### 3.1.1 This is heading 3 style

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

1. The quick brown fox jumps over the lazy dog.
2. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
3. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

* The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
* The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
* The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

## Unnumbered list style

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

 (1)

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

**Fig. 1. Relation between τ and *t*.** This example has only two continuous Steppers, S1 and S2.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

**Table 1.**Benchmark results of the cascade oscillators model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |S| | Predicted cost | Timing | Predicted speed | Speed |
| 1 | S219.20(100%) | 68m43s | 1.00 | 1.00 |
| 2 | 29.10+219.10(~50%) | 35m13s | 2.00 | 1.95 |
| 4 | 219.20(100%) | 68m43s | 1.00 | 1.00 |
| 10 | 29.10+219.10(~50%) | 35m13s | 2.00 | 1.95 |
| 20 | 219.20(100%) | 68m43s | 1.00 | 9.5 |

This is table foot note sample text This is table foot note sample text This is table foot note sample text

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog.

Acknowledgements

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

References

(1) Weinreb, C.; Riesselman, A. J.; Ingraham, J. B.; Gross, T.; Sander, C.; Marks, D. S. 3D RNA and Functional Interactions from Evolutionary Couplings. *Cell* **2016**, *165*, 963–975.

(2) Henikoff, S.; Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America* **1992**, *89*, 10915–10919.

(3) Nielsen, H.; Engelbrecht, J.; Brunak, S.; Heijne, G. von. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein engineering* **1997**, *10*, 1–6.

(4) Nielsen, H.; Krogh, A. Prediction of signal peptides and signal anchors by a hidden Markov model. *Proceedings. International Conference on Intelligent Systems for Molecular Biology* **1998**, *6*, 122–130.

(5) Laroum, S.; Duval, B.; Tessier, D.; Hao, J.-K. A Genetic Algorithm for Scale-Based Translocon Simulation. In *Pattern Recognition in Bioinformatics*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Shibuya, T., Kashima, H., Sese, J., Ahmad, S., Eds.; Lecture Notes in Computer Science; Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp 26–37.

(6) Almagro Armenteros, J. J.; Tsirigos, K. D.; Sønderby, C. K.; Petersen, T. N.; Winther, O.; Brunak, S.; Heijne, G. von; Nielsen, H. SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nature biotechnology* **2019**, *37*, 420–423.

(7) Heinzinger, M.; Elnaggar, A.; Wang, Y.; Dallago, C.; Nechaev, D.; Matthes, F.; Rost, B. *Modeling the Language of Life - Deep Learning Protein Sequences* 360, 2019.

(8) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9*, 1735–1780.

(9) Peters, Matthew E., Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, Zettlemoyer, Luke. *Deep contextualized word representations*, 2018.

(10) Hayat, S.; Sander, C.; Marks, D. S.; Elofsson, A. All-atom 3D structure prediction of transmembrane β-barrel proteins from sequences. *Proceedings of the National Academy of Sciences of the United States of America* **2015**, *112*, 5413–5418.

(11) Marks, D. S.; Hopf, T. A.; Sander, C. Protein structure prediction from sequence variation. *Nature biotechnology* **2012**, *30*, 1072–1080.

(12) Suzek, B. E.; Wang, Y.; Huang, H.; McGarvey, P. B.; Wu, C. H. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics (Oxford, England)* **2015**, *31*, 926–932.

(13) Steinegger, M.; Meier, M.; Mirdita, M.; Voehringer, H.; Haunsberger, S. J.; Soeding, J. *HH-suite3 for fast remote homology detection and deep protein annotation*, 2019.

(14) Perdigão, N.; Heinrich, J.; Stolte, C.; Sabir, K. S.; Buckley, M. J.; Tabor, B.; Signal, B.; Gloss, B. S.; Hammang, C. J.; Rost, B.; Schafferhans, A.; O'Donoghue, S. I. Unexpected features of the dark proteome. *Proceedings of the National Academy of Sciences of the United States of America* **2015**, *112*, 15898–15903.

(15) Schafferhans, A.; O'Donoghue, S. I.; Heinzinger, M.; Rost, B. Dark Proteins Important for Cellular Function. *Proteomics* **2018**, *18*, e1800227.

(16) UniProt Consortium, T. UniProt: the universal protein knowledgebase. *Nucleic acids research* **2018**, *46*, 2699.

(17) Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. Automatic differentiation in PyTorch. [Online] **2017**.

(18) Tompson, Jonathan, Goroshin, Ross, Jain, Arjun, LeCun, Yann, Bregler, Christopher. *Efficient Object Localization Using Convolutional Networks*, 2014.

(19) Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Online] **2014**.

(20) Ioffe, Sergey, Szegedy, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015.

(21) Kingma, Diederik P., Ba, Jimmy. *Adam: A Method for Stochastic Optimization*, 2014.