ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #44652

Towards Streamlining the Process of Building Machine Learning Models for your Artificial Intelligence Applications

Mr. Joseph George, Western Michigan University
Dr. Ajay Gupta, Western Michigan University
Alvis Fong, Western Michigan University

Towards Streamlining the Process of Building Machine Learning Models for your Artificial Intelligence Applications

Joseph George
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008
Email : joseph.77.george@wmich.edu

Ajay Gupta
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008
Email : ajay.gupta@wmich.edu

Alvis C Fong
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008
Email : alvis.fong@wmich.edu

Abstract: Rapid recent advances in Artificial Intelligence (AI)-based tools in diverse disciplines have created a need to develop educational material for AI-readiness of the workforce. In this work, we focus on the problem of designing and developing machine learning (ML) models with ease. This paper thus undertakes an investigation into the automatic development of machine learning models with minimal user expertise through the use of AutoKeras, an automatic ML python library. AutoKeras streamlines the typically intricate ML development process which traditionally demanded the expertise of ML engineers. This paper will first walk through the typical ML model development process. After this process is understood, AutoKeras' role in making this process simpler and more accessible will be discussed and showcased with an application to the proteomics domain.

I. Introduction

In the conventional ML project workflow, navigating through pre- and post-processing steps such as: data cleaning, balancing, and fine-tuning ML models poses a considerable challenge, often requiring substantial technical and domain expertise. Acknowledging this hurdle, this paper discusses novel methodologies such as Neural Architecture Search (NAS)[1], Efficient Neural Architecture Search (ENAS)[2], and the Google AutoML/AutoKeras[4] libraries as

accessible alternatives that ease the ML development process. While these methodologies have proven successful in image and speech classification, their application in other disciplines such as the genome[6] and proteomics[6] domain remains largely unexplored. In proteomics, the function and interactions of proteins within a biological system are studied, providing a clearer understanding of the structure and function of organisms than genomics[8]. Traditionally manually curated models have found success in proteomics through applications such as de novo sequencing[3] and peptide analysis[3]. This paper exhibits the viability of AutoKeras as a competitor to manually curated models for those with limited ML knowledge. We use the challenging problem of amino acid identification[3] to demonstrate the ease of automatically developing deep-learning models to identify amino acids from the mass-spectrometry data.

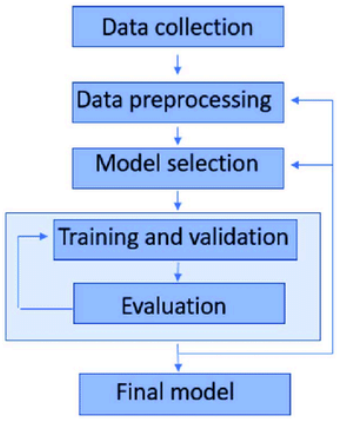

Fig 1. Machine Learning Model Development Process[9]

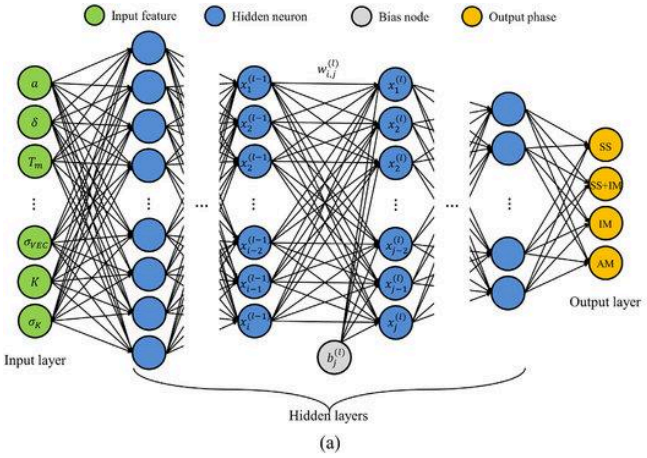

Fig. 2 A sample Deep Neural Network model[7]

Fig. 1[9] shows a step-by-step breakdown of the general process for manually developing ML models for your AI-applications. After identifying the problem where ML models can aid in

the prediction for your business use case, the first step is data collection. During this phase, the developer needs to find the appropriate dataset which can either be collected or found in repositories such as the Google dataset repository[10] to train the model on. There are many ongoing efforts by government agencies, professional societies and private enterprises to build such repositories for various domains. It is important that this dataset is large enough and varied enough for the model to be able to effectively generalize the information presented in the dataset for supervised learning (a reader can refer here[11], for example, to understand the properties and characteristics of data.) It is also important for this dataset to be "labeled" correctly. This is important because in order for the model to learn what the correct output is, it needs to know the starting point (the training data) and the ending point (the labeled output) to derive intermediate layers and then apply that to the data it has not seen before. Usually when a model has to be trained, the available dataset is split into two categories - the training set and the validation set. The models are then trained on the training set and the final model is tested on the validation set. This is an important step so as to know the performance of the model on an unfamiliar dataset.

The next step is data preprocessing. Data preprocessing typically manipulates the data in a way that makes the training of the model more efficient and makes the model more effective and accurate. This phase includes the elimination of noise in the dataset. In this case, "noise" refers to insignificant background information that is unnecessary for the model to train on. It is important to note that a trained ML engineer would need to decide how to best carry out this step as what is considered noise varies for each domain. There could also be other steps involved in preprocessing such as normalization, balancing etc. More about the different preprocessing methods can be found, for example, in this guide[12].

After the data is preprocessed, a model type is chosen by the engineer. The model type depends on the type of data that the model outputs and also changes how the model learns such as the example shown in Fig. 2[7]. Because of this, the model types vary from application to application. For a Deep Neural Network, such as in this experiment, it would mean determining the number of initial features, layers of nodes required, optimizer, objectives etc. Different model types are briefly discussed, for example at MathWorks[13].

The ML engineer then trains the model after an appropriate model type is selected. The model also provides an assessment metric, such as accuracy, sensitivity, specificity etc., which is based on a comparison between the values that the model found and the correct output values that is associated with that part of the dataset.

Next comes the most time consuming part for the engineer which is evaluation. During this step, which step to revisit in an attempt to increase the model's performance. The engineer may try to preprocess the data in different ways, select a different model type, or manually tell the model what parts of the data is the most important to focus on.

When the ML engineer has gone through a sufficient number of iterations to get the desired accuracy, the final model has been found and is saved.

II. The amino acid identification application

Preprocessing typically depends on the domain. In our example proteomics application, spectrometry data is used to train the model to identify amino acids in the sample. An MS/MS spectrum can be represented as a histogram of intensity vs. mass-to-charge ratio of the ions acquired from the peptide fragmentation within a mass spectrometer[5]. Different types of ions may have different intensity values, seen as peak heights on the spectra, depending on the fragmentation methods. In the spectrum there are also many noise peaks, that should be removed, that are mixed in with the real ions. The spectrometry data is usually stored in RAW or MGF file formats and requires conversion of these file formats to pickle format (Why Pickle format?[14]).

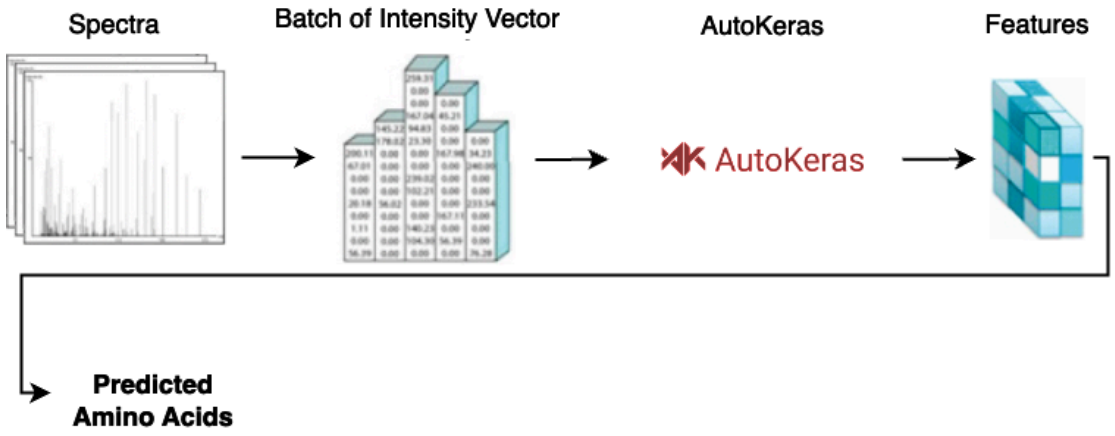

Fig. 3 Amino Prediction Using AutoKeras

For this experiment, the MS/MS spectra data was collected from the PRIDE Peptidome library[15]– a proteomics repository with large labeled datasets. During data preprocessing, the data was balanced, scaled and labeled before AutoKeras used it for training. AutoKeras is then configured with an objective function to pick a model and search for an optimal model after multiple iterations of training different models and improving the accuracy to identify the amino acids for a particular species.

During the process of engineering a model, ML engineers would typically manually tune the parameters of the model, such as the initial weights to the features, deciding the

number of layers, optimizer etc. This is tedious as it requires manual intervention at each step of the process. However, much of this process can be automated with the use of AutoKeras. This library allows the user to generate high-performing models with minimal intervention from the user. In the case of structured data classification, the user can provide AutoKeras with the input data, and it automatically develops the optimal model for the dataset (see Figure 3). Steps from model selection to evaluation in a traditional ML pipeline (see Figure 1) are automated when AutoKeras is employed. AutoKeras leverages Keras and TensorFlow for model construction, utilizing KerasTuner for hyperparameter tuning, thus offering user-friendly APIs. Its workflow comprises data analysis, construction of a search space based on the data, and the use of a search algorithm to identify optimal model configurations. AutoKeras incorporates cutting-edge models like EfficientNet and BERT, while also tuning hyperparameters for preprocessing and training steps.

```
# define the search
search = StructuredDataClassifier(overwrite = True, max_trials = 1, objective = "val_accuracy")
# perform the search
search.fit(x = x_train, y = y_train, validation_data = (x_test, y_test), verbose = 1, epochs = 50)
```

Fig. 4 Using AutoKeras to Search for the Optimal Model

As demonstrated in Figure 4, AutoKeras makes the development of quality models quick and simple for the user. Compared to manually curated models, AutoKeras allows the user to seamlessly transition from preprocessing the input data to training and evaluating the model. In addition to its simplicity, AutoKeras grants users the flexibility to configure various aspects of the search process when using the Structured Data Classifier class (see Figure 5). For new users, this ability to effortlessly control search factors such as the number of epochs, the percentage of validation data to use, and the use of multiprocessing enables fine-tuning and ensures that the resulting models meet specific requirements and performance benchmarks. Moreover, more advanced users can further customize the search space by utilizing the AutoModel class to define the high-level architecture of the model (see Figure 6).

```
StructuredDataClassifier.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose='auto',
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False
)
```

Fig. 5 Parameters of Structured Data Classifier Fit Method

```
# The user specifies the high-level architecture.
import autokeras as ak

image_input = ak.ImageInput()
image_output = ak.ImageBlock()(image_input)
text_input = ak.TextInput()
text_output = ak.TextBlock()(text_input)
output = ak.Merge()([image_output, text_output])
classification_output = ak.ClassificationHead()(output)
regression_output = ak.RegressionHead()(output)
ak.AutoModel(
    inputs=[image_input, text_input],
    outputs=[classification_output, regression_output]
)
```

Fig. 6 Customizing the Search Space with AutoKeras

AutoKeras' ease of use coupled with its versatility enables it to compete as a viable alternative to manual ML, simplifying training models by streamlining the Neural Architecture Search (NAS) process. It implements Reinforcement Learning[16] techniques, to guide the search for optimal architectures by making decisions about what architectural components to add to the neural network. This library samples child models from the search space and automatically trains them. Using techniques such as cross-validation or holdout validation[17] on a validation set and through the use of reinforcement learning algorithms such as policy gradients[18] and samples from child models, AutoKeras estimates the performance of the child model. It then updates the search strategy, allowing it to develop a policy for finding promising architectures to explore further. Once the search is completed, AutoKeras selects the final architecture based on the performance estimates obtained during the search. AutoKeras then trains the architecture

on the full training data for the number of epochs specified by the user to develop the final model. After AutoKeras trains the optimal model, the user evaluates it on a validation dataset to calculate its accuracy[4]. More documentation on AutoKeras setup can be found here[19].

III. Experiment details

This experiment showcases how AutoKeras can be used to obtain a model that is competitive with manually curated models with little to no knowledge on machine learning. Our experiment was designed so one can compare our results with the currently popular publicly available peptide identification models like DeepNovo[3]. It uses the mass spectrometry data to predict whether the given amino acid is present in the peptide or not. Although our model was created with manually preprocessed data and the manual selection of model type, AutoKeras chose the features and the model architecture that an AI engineer would typically choose. For the Escherichia coli dataset, DeepNovo model reported an amino acid identification precision of 52.3%[3], whereas our amino acid identification models had an average amino acid identification accuracy of 68.03%. This is after generating 20 models, each for an amino acid present in Escherichia Coli. More details on the identification accuracy and time required to generate the model can be found in Table 1. While the preprocessing of data remained the same between manual curation of model and the one using AutoKeras, the experiment using AutoKeras was completed way faster in determining what would be the ideal model for amino acid identification as designing the model manually required many days of refining the model design which requires expertise and AutoKeras just required over a day to train and save the model for 20 amino acids of Escherichia Coli species. This experiment with AutoKeras demonstrates how artificial Neural Networks (ANNs) can be effectively implemented in a new domain with minimal expertise and hence save time in defining the accurate model for the data chosen. The experiment was run on a Michigan State University cluster having one NVIDIA A100 GPU, Intel XEON CPU with 36GB of allocated memory.

Amino acid	Accuracy	Time duration
A	0.706	00h 47m 06s
R	0.564	01h 53m 16s
N	0.654	03h 07m 00s
D	0.651	01h 31m 36s
C	0.939	01h 28m 31s
Q	0.625	01h 18m 01s

E	0.647	01h 01m 53s
G	0.648	00h 43m 14s
H	0.764	01h 57m 20s
I	0.592	03h 00m 48s
L	0.755	01h 15m 17s
K	0.584	01h 47m 09s
M	0.749	01h 54m 56s
F	0.628	02h 27m 51s
P	0.624	00h 44m 26s
S	0.608	01h 41m 17s
T	0.620	00h 40m 59s
W	0.894	01h 43m 11s
Y	0.692	01h 36m 44s
V	0.652	01h 16m 20s

Table 1. Amino acid identification accuracy in E. Coli

IV. Outcome assessment

We plan to use the AutoML process discussed in this paper in the Artificial Intelligence and Machine Learning courses offered in 2024 for problems arising from two different domains. Learning activities both homework as well as in-class experiential learning that support the theory students learn will be fully integrated to assess the effectiveness of the educational module. Feedback will be obtained for continual improvement using questionnaires that conform to pedagogical and andragogical literature from the evaluation community.

V. Conclusion

We exhibited a workflow that facilitates easier development of efficient ML models by a user with minimal AI/ML expertise using AutoKeras, an AutoML python library. A comparison between DeepNovo, a manually curated model, and the AutoKeras model demonstrates the promise of using AutoKeras to develop the best-performing model for different datasets. This effectiveness shows how those with minimal knowledge of ANN model development are able to create highly specific models that can compete with and even outperform hand-tailored models relevant to their domain.

Acknowledgement

This work was supported in part by the National Science Foundation under Grant OAC-2320951, National Institute of Health under Grant1R15GM120820-01A1 and WMU FRACAA 2021-22. We thank Quinn Hague, Brady Jensenius and Justin Rosales from the Kalamazoo Area Mathematics and Science Center for their support during the experiments.

Bibliography

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[2] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *CoRR*, vol. abs/1802.03268, 2018. [Online]. Available: http://arxiv.org/abs/1802.03268

[3] N. H. Tran, X. Zhang, L. Xin, B. Shan, and M. Li, "De novo peptide sequencing by deep learning," *Proceedings of the National Academy of Sciences*, vol. 114, pp. 8247 – 8252, 2017.

[4] H. Jin, F. Chollet, Q. Song, X. Hu, "AutoKeras: An AutoML Library for Deep Learning," *Journal of Machine Learning Research*, vol. 24, 2023.

[5] J. S. Cottrell, "Protein identification using ms/ms data." *Journal of proteomics*, vol. 74 10, pp. 1842–51, 2011.

[6] N. Saraswathy and P. Ramalingam, *Concepts and Techniques in Genomics and Proteomics*. Hexagon House, Avenue 4, Station Lane, Witney, Oxford OX28 4BN, UK: Biohealthcare Publishing (Oxford) Limited, 7 2011, vol. 10, ch. 12, pp. 171–183.

[7] Lee, Soo-Young & Byeon, Seokyeong & Kim, Hyoung & Jin, Hyungyu & Lee, Seungchul. (2020). "Deep learning-based phase prediction of high-entropy alloys: Optimization, generation, and explanation. Materials and Design". 197. 10.1016/j.matdes.2020.109260.

[8] Al-Amrani, Safa et al. "Proteomics: Concepts and applications in human medicine." *World journal of biological chemistry* vol. 12,5 (2021): 57-69. doi:10.4331/wjbc.v12.i5.57

[9] Ng, Frederick & Jiang, Ruihan & Chow, James. (2020). "Predicting radiation treatment planning evaluation parameter using artificial intelligence and machine learning". IOP SciNotes. 1. 014003. 10.1088/2633-1357/ab805d.

[10] Google Research Datasets. (n.d.). https://research.google/resources/datasets/

[11] Anand. (2019, March 31). "*Why domain knowledge important in data science?"*. Medium.https://medium.com/@anand0427/why-domain-knowledge-is-important-in-data-science-anand0427-3002c659c0a5

[12] Ghosh, S. (2023, August 22). "*A comprehensive guide to data preprocessing"*. Neptune.ai. https://neptune.ai/blog/data-preprocessing-guide

[13] Types *of machine learning models explained*. Types of Machine Learning Models Explained - MATLAB & Simulink. (n.d.). https://www.mathworks.com/discovery/machine-learning-models.html

[14] Jain, A. N. (2020, May 30). "*Why turn into a Pickle?"*. Medium. https://towardsdatascience.com/why-turn-into-a-pickle-b45163007dac

[15] Pride - Proteomics Identification Database. (n.d.). https://www.ebi.ac.uk/pride/peptidome

[16] MIT OpenCourseWare. (n.d.). *Lecture 16: Reinforcement learning, part 1: Machine learning for healthcare: Electrical engineering and computer science*. MIT OpenCourseWare. https://ocw.mit.edu/courses/6-s897-machine-learning-for-healthcare-spring-2019/resources/lecture-16-reinforcement-learning-part-1/

[17] Cross validation. (n.d.). https://www.cs.cmu.edu/~schneide/tut5/node42.html
[18] Kapoor, S. (2018, June 2). "*Policy gradients in a Nutshell"*. Medium. https://towardsdatascience.com/policy-gradients-in-a-nutshell-8b72f9743c5d
[19] *Autokeras*. AutoKeras. (n.d.). https://autokeras.com/