Automating the Future: A Comprehensive Review of Machine Learning Pipeline Generation

Yutao Chen Hieu Nguyen Minh
School of Informatics, Universitat Politècnica de Catalunya

1. Introduction

Despite the advancements in Machine Learning (ML), the creation and finetuning of ML pipelines remain challenging and requires domain expertise. Automating ML pipelines can significantly reduce the time and expertise needed, leading to the concept of automated ML (AutoML).

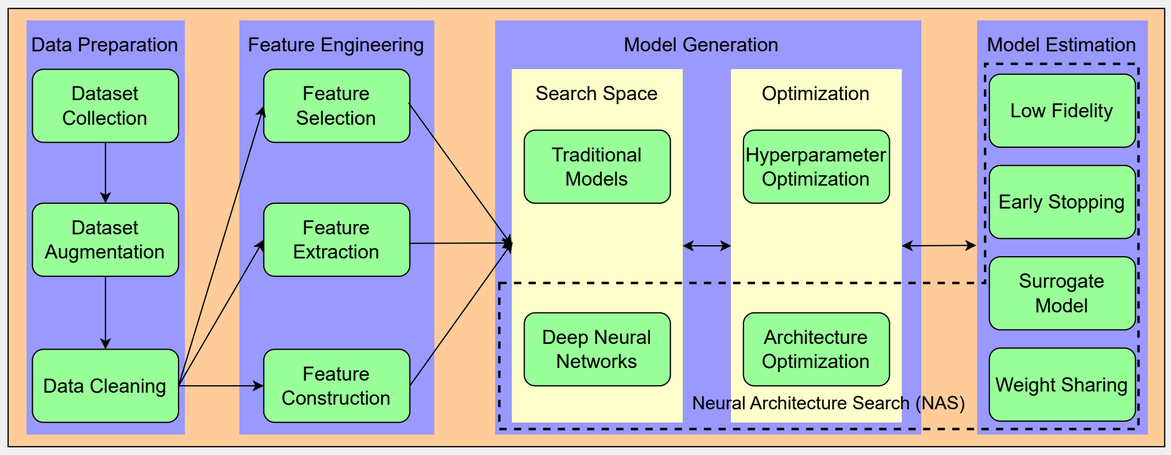

Figure 1. AutoML pipeline (reproduced from [1]).

An AutoML pipeline (Figure 1) consists of four main steps [1]:

1. **Data Preparation**: Preparing a clean dataset from a raw one, probably adding augmented data.
2. **Feature Engineering**: Creating features to feed the model.
3. **Model Creation**: Creating an optimized model in two steps:
   - **Search Space**: Choosing the model type, which can be a traditional model or a deep neural network.
   - **Optimization**: Containing *HyPerparameter Optimization (HPO)* and *Architecture Optimization (AO)*
4. **Model Estimation**: Evaluating the model performance.

The search space of neural networks, the AO, along with the model estimation methods, form the *Neural Architecture Search (NAS)*.

2. Pipeline Creation Problem

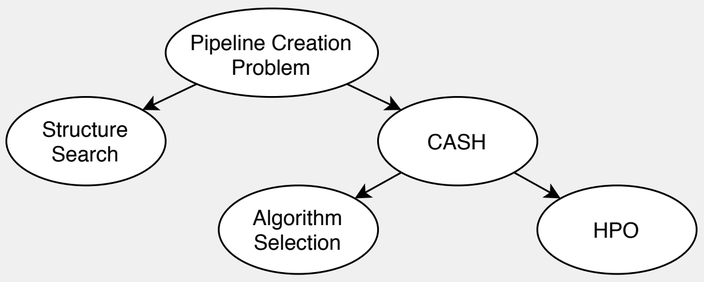

Figure 2. Subproblems of the pipeline creation problem [2].

The state-of-the-art algorithms that solve the pipeline creation problem are categorized into Structure Search (an instance is NAS) and *Combined Algorithm Selection and Hyperparameter* (CASH) problem (Figure 2) [2]. Let $\mathcal{L}\left(A_{\boldsymbol{\lambda}}^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)}\right)$ be the loss that algorithm $A^{(j)}$ achieves on $D_{valid}^{(i)}$ when trained on $D_{train}^{(i)}$ with hyperparameters $\boldsymbol{\lambda}$. CASH finds the joint algorithm and hyperparameter setting that minimizes that loss:

$$A^{\star}, \boldsymbol{\lambda}_{\star} \in \underset{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \mathcal{L}\left(A_{\boldsymbol{\lambda}}^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)}\right). \quad (1)$$

3. Data Preparation

Figure 3 shows the flow chart of data preparation. The first step is to obtain a raw dataset, which can be a provided one or an extended one with data augmentation. Then, data cleaning is used to handle noisy and missing data. A dataset can further be improved with augmented data, which can potentially enhance model performance and robustness.

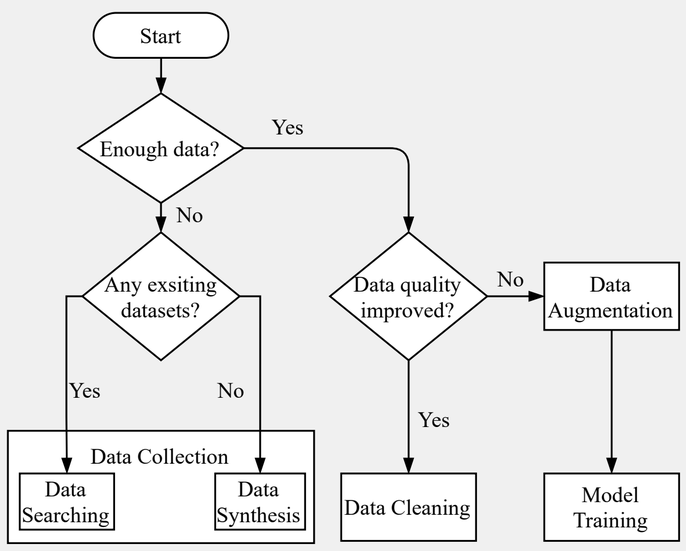

Figure 3. Data preparation flow chart [1].

4. Feature Engineering

Feature Selection minimizes feature redundancy by identifying the most important features.

Feature extraction reduces the dimensionality of features through specific mapping functions.

Feature Construction expands the original feature space with preprocessing transformation.

5. Model Generation

Traditional Models includes Support Vector Machine (SVM), k-nearest neighbors algorithm (KNN), and Perceptron Learning Algorithm (PLA).

Deep Neural Networks SOTA search space methods include *Efficient Neural Architecture Search (ENAS)* [3], *Cell-based approach* [4], *Hierarchical search space* [5], *Morphism-based method* [6]. They are instances of NAS technique which has recently attracted lots of attention.

Hyperparameter Optimization SOTA methods includes grid search/random search and Bayesian/gradient-based optimization.

Architecture Optimization SOTA methods includes *Evolutionary Algorithm (EA)* [7], *Reinforcement Learning (RL)* [3], *Differentiable Architecture Search (DARTS)* [8], and surrogate model-based optimization [9].

6. Model Evaluation

Low fidelity methods provide an approximate evaluation without requiring full training and evaluation cycles.

$$\text{Resource Saving} = \frac{\text{Resources for Full Dataset}}{\text{Resources for Proxy Dataset}}. \quad (2)$$

Weight sharing allows multiple architectures to share a common set of weights for certain layers, reducing the number of training parameters.

A surrogate model is a cheaper-to-evaluate approximation of the true performance function of a machine learning algorithm with respect to its hyperparameters.

Early stopping is a regularization technique used to prevent overfitting by stopping the training if the model performance does not improve for a specified number of epochs.

7. Conclusion

AutoML reduces the demand for data scientists by enabling domain experts on ML applications without extensive knowledge of ML. The progress made so far is promising, and our review serves as a resource for researchers to foster further advancements in AutoML pipelines.

References

[1] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-based systems*, 2021.

[2] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," *Advances in neural information processing systems*, vol. 28, 2015.

[3] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[4] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018.

[6] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. ACM, 2016.

[7] L. Xie and A. L. Yuille, "Genetic cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2017.

[8] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[9] R. Negrinho, M. R. Gormley, G. J. Gordon, D. Patil, N. Le, and D. Ferreira, "Towards modular and programmable architecture search," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 13 715–13 725.