Introduction to AutoML

AutoML. A set of very versatile tools to automate the machine learning process end to end. e.
You'll be learning about **search spaces** and **strategies** which are key for both hyper parameter tuning and AutoML.
We'll also discuss tools to **quantify performance estimation** on the explored models in your search.
Finally, you learn about **different AutoML offerings** available in the Cloud by major service providers.
It covers the complete pipeline from the **raw data** set to the **deployable** machine learning model.

Neural Architecture Search or **NAS** is at the **heart** of AutoML.
There are three **main parts** to Neural Architecture Search,
a search space:

- the range of architectures which can be represented. To **reduce the size of the search problem,** we need to limit the search space to the architectures which are best suited to the problem that we're trying to model. This helps reduce the search space, but it also means that **a human bias** will be introduced, which might prevent Neural Architecture Search from finding architectural blocks that go beyond current human knowledge.

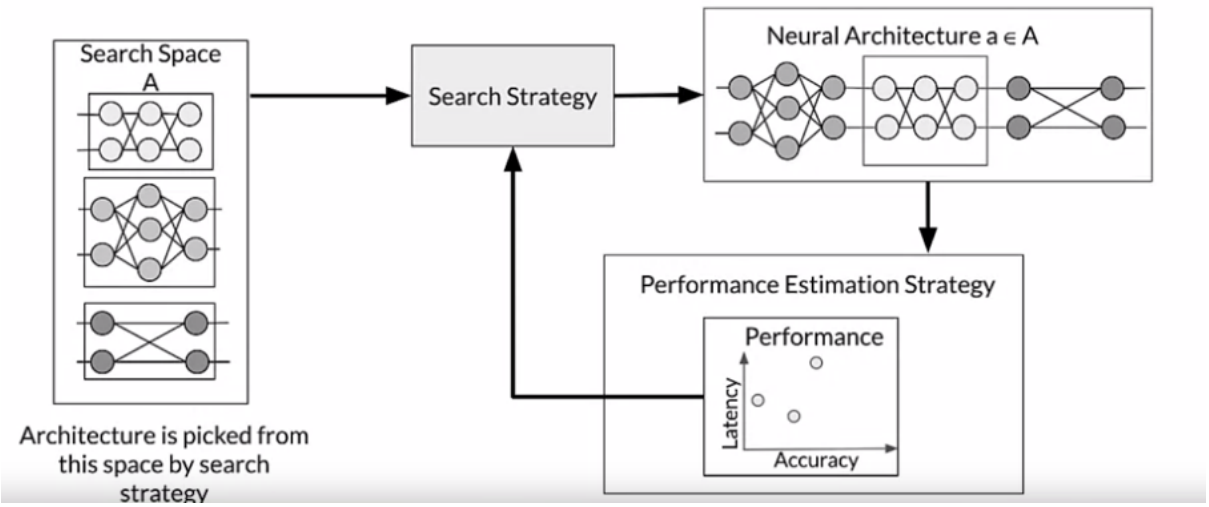
a search strategy:

- how we explore the search space. We want to explore the search based **quickly**, but this might lead to **premature convergence to a sub optimal region in the search space**. The objective of Neural Architecture Search is to find **architecture is that perform well on our data**.

a performance estimation strategy:

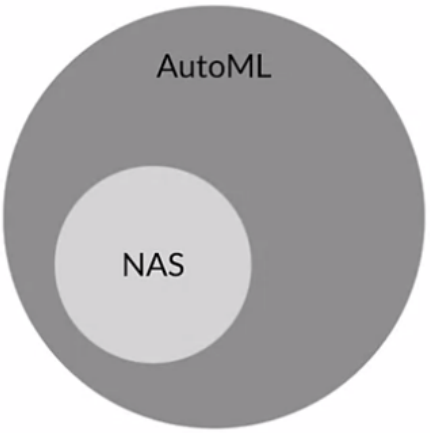
- helps in measuring and comparing the performance of various architectures
- A search strategies selects an architecture from a predefined search space of architectures. The selected architecture is passed to a performance estimation strategy, which returns it's estimated performance to the search strategy.

Neural Architecture Search



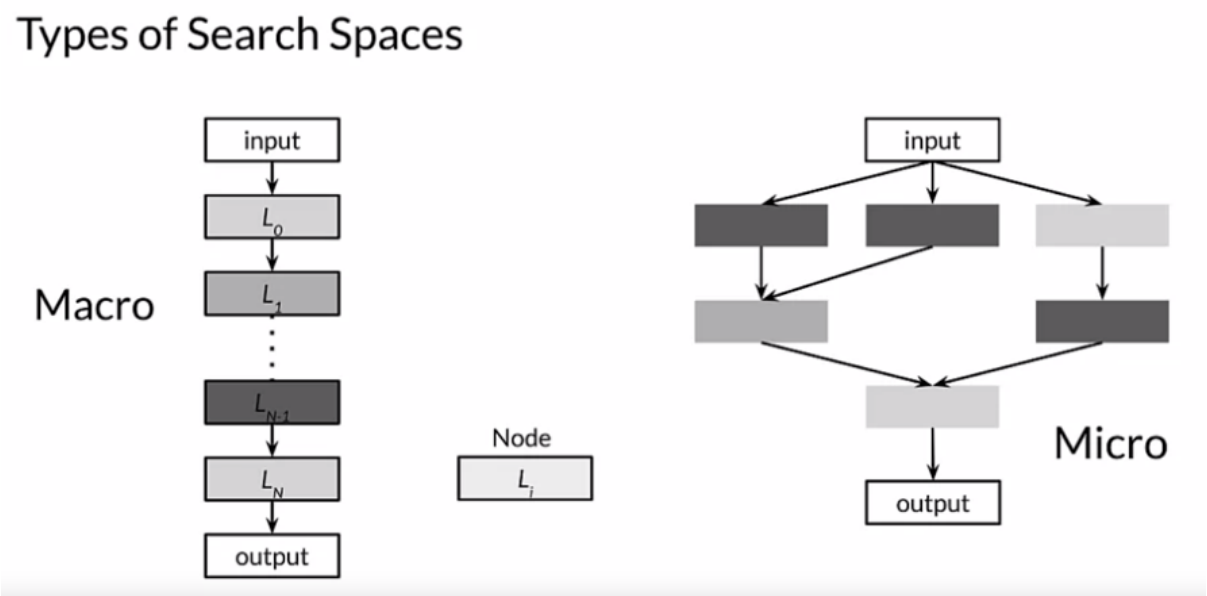
Neural Architecture Search

- **AutoML** automates the development of ML models
- **AutoML** is not specific to a particular type of model.
- Neural Architecture Search (**NAS**) is a subfield of AutoML
- NAS is a technique for automating the design of artificial neural networks (ANN).



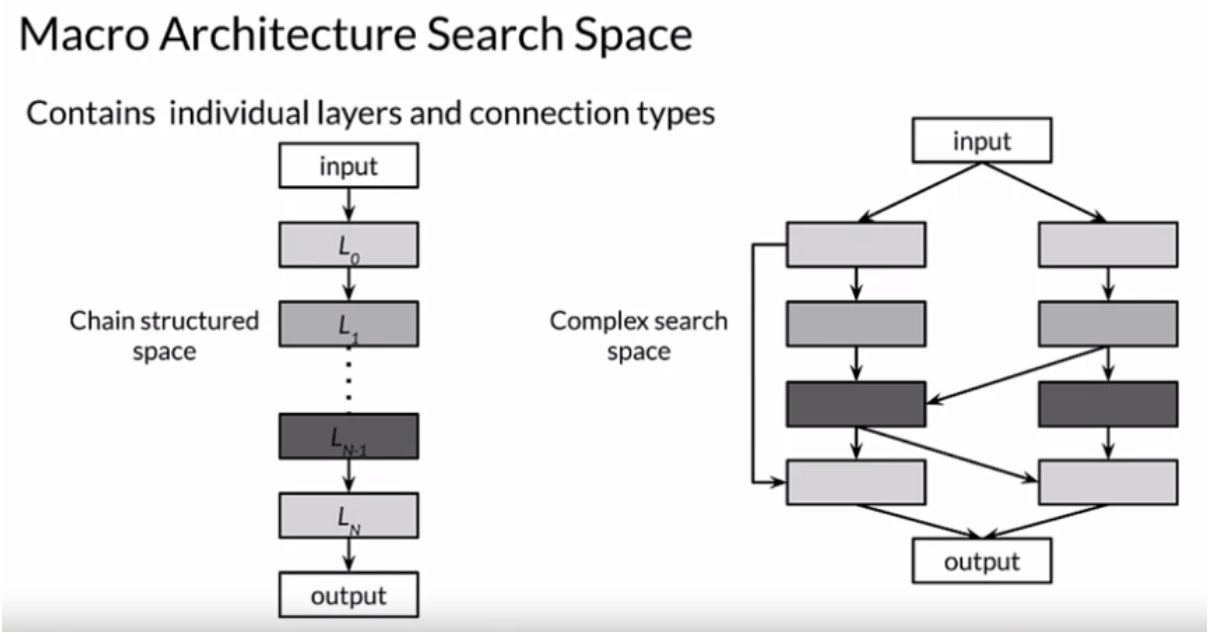
Understanding Search Spaces

search spaces, **macro** and **micro**.



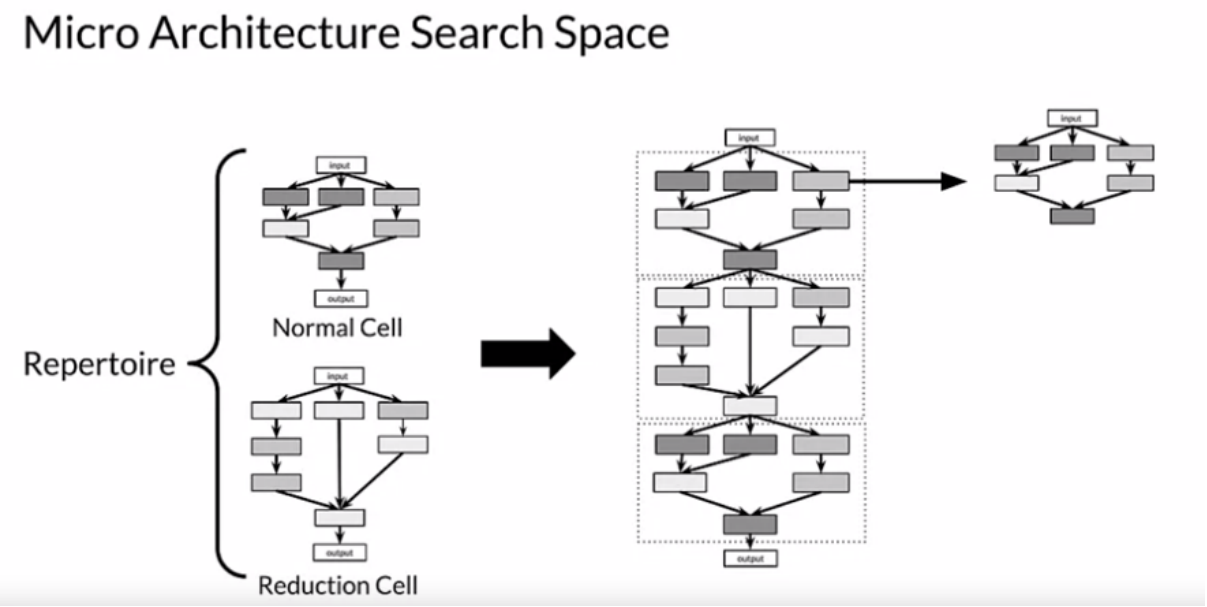
MACRO:

- A macro search space contains the individual layers and connection types of a neural network. And neural architecture search searches within that space for the best model, building the model layer by layer
- a network can be built very simply by stacking individual layers referred to as a chain structured space or with multiple branches and skip connections in a complex space



MIRCO:

- neural architecture search builds a neural network from cells where each cell is a smaller network. Here are two different types of cells, a normal cell on the top and a reduction cell on the bottom. Cells are stacked to produce the final network. This approach has been shown to have s



ignificant performance advantages compared to a macro approach.

Search Strategies

Neural architecture search searches through the search base **for the architecture** that produces the best performance.
A variety of different **approaches** can be used for that search:

- grid search
    - search everything, that means you cover every option we have in the search base
- random search
    - you select the next option randomly within the search space

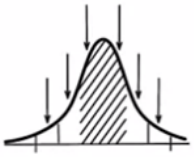
for grid and random search:

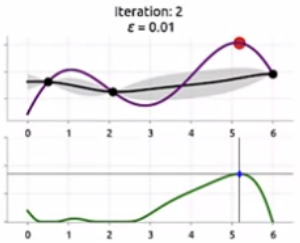
- Both of these work reasonably well **in smaller search bases**, but both also fail fairly quickly. When the search space grows beyond a certain size, which is all too common.

- Bayesian optimization
    - a little more sophisticated. It assumes that a specific probability distribution, which is typically a Gaussian distribution is underlying the performance of model architectures. So you use observations from tested architectures to constrain the probability distribution and guide the selection of the next option.This allows us to build up an architecture stochastically based on the test results and the constrained distribution.

Bayesian Optimization

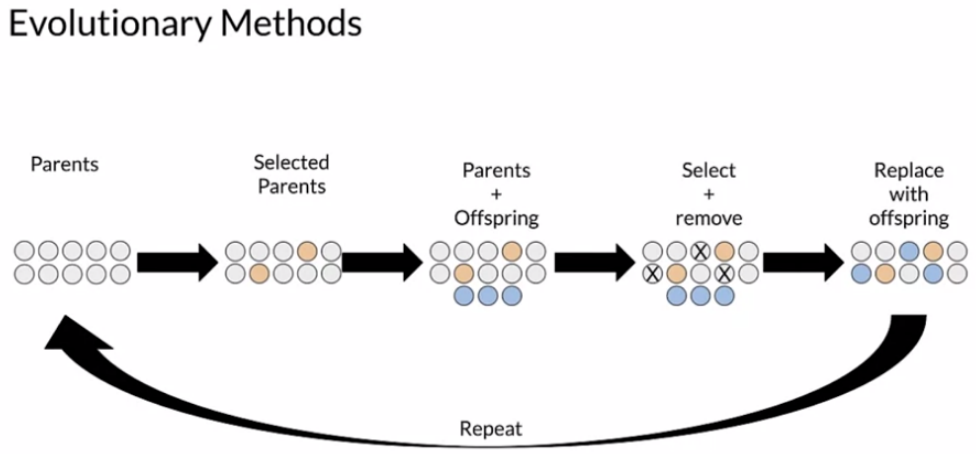
- Assumes that a *specific probability distribution*, is underlying the performance.
- Tested architectures constrain the probability distribution and guide the selection of the next option.
- In this way, promising architectures can be stochastically determined and tested.





-

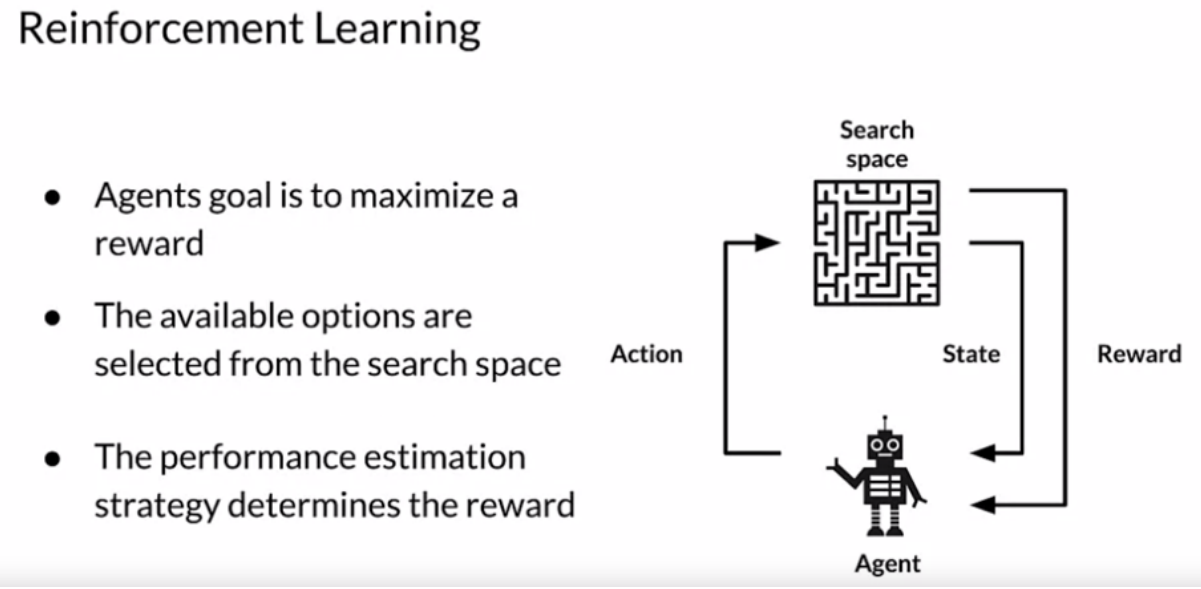
- evolutionary algorithms
  - First, an initial population of n different model architecture is **randomly generated,** the performance of each individual. In other words, each architecture is evaluated as defined by the performance estimation strategy, which we'll talk about next. Then the X highest performers are selected as parents for **a new generation.** This new generation of architectures might be copies of the respective parents with induced random **alterations** or **mutations**. Or they might arise from combinations of the parents, the performance of the offspring is assessed.
  - Again using the performance estimation strategy. The list of possible **mutations** can include operations like adding or removing a layer. Adding or removing a connection, changing the size of a layer or changing another hyper parameter. Y architectures are selected to be removed from the population. This might be the Y worst performers, the Y oldest individuals in the population. Or a selection of individuals based on a combination of these parameters. The offspring replaces the removed architectures and the process is restarted with this new population.



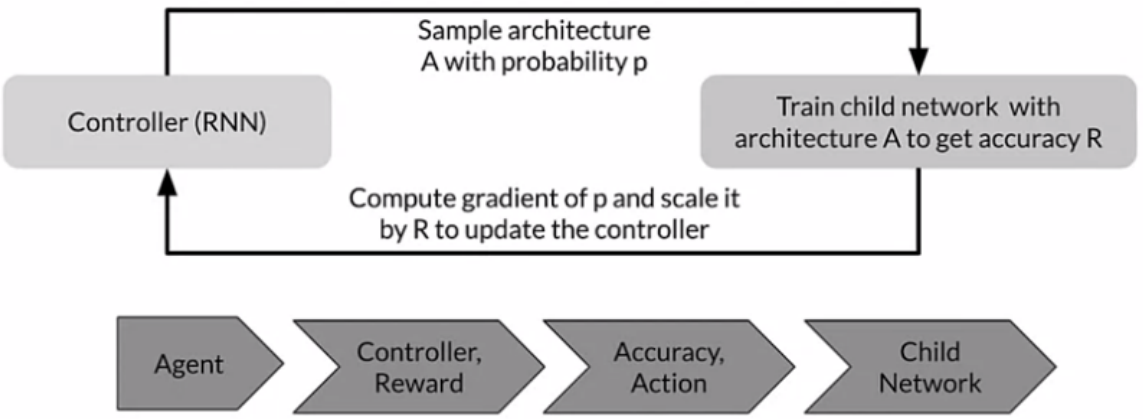
-

- reinforcement learning.
  - agents take actions in an environment, trying to maximize a reward. After each action, the state of the agent and the environment is updated and a reward is issued based on a performance metric. Then the range of possible next actions is

evaluated, the environment in this case is our search space. And the reward function is our performance estimation strategy.



- A neural network can also be specified by a variable length string where the elements of the string specify individual network layers. That enables us to use a recurrent neural network or RNN to generate that string, as we might do for an NLP model. The RNN that generates the string is referred to as the controller, after training the network referred to as the child network on real data. We can measure the accuracy on the validation set, the accuracy determines the reinforcement learning reward in this case. Based on the accuracy, we can compute the policy gradient to update the controller RNN. In the next iteration, the controller will have learned to give higher probabilities to architectures that result in higher accuracies during training.



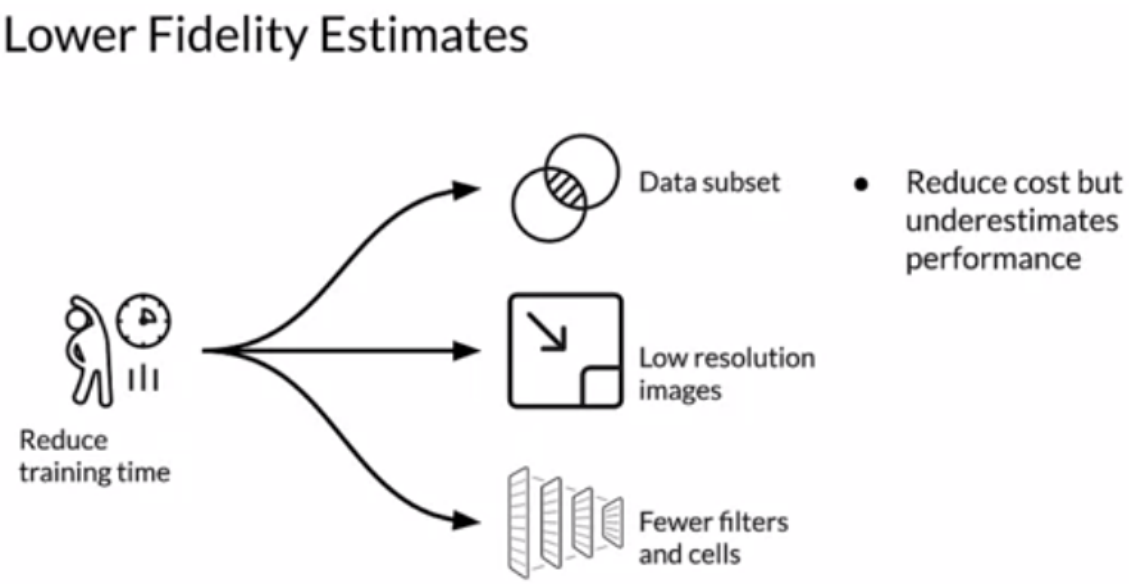
-

https://distill.pub/2020/bayesian-optimization/
and other papers for further reading

Measuring AutoML Efficacy

measure the accuracy or effectiveness of the different architectures that it tries. This requires a performance estimation strategy.
The simplest approach is to measure the **validation accuracy** of each architecture that is generated like we saw with the reinforcement learning approach. This becomes computational **heavy**, especially for large search spaces and complex networks. And as a result, it can take several GPU days to find the best architectures using this approach, that makes it **expensive** and **slow**. It also makes neural architecture search impractical for many use cases.
Other approaches:

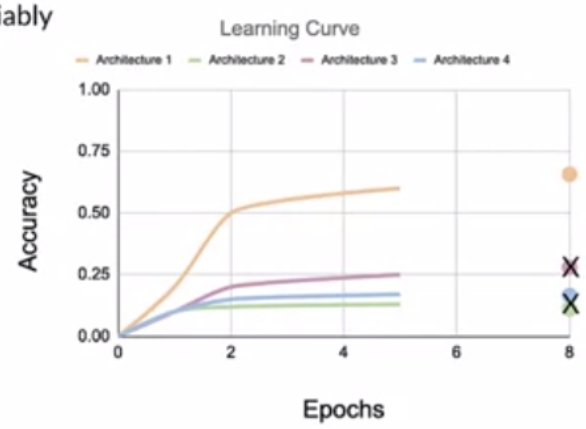
- lower fidelity 逼真度 estimates
    - Lower fidelity or lower precision estimates try to **reduce the training time** by reframing the problem to make it easier to solve by training on a subset of data or using lower resolution images, for example, or using fewer filters per layer and fewer cells. It greatly reduces the computational cost, but ends up underestimating performance. That's okay if you can make sure that the relative ranking of the architectures does not change due to their lower fidelity estimates. But unfortunately, recent research has shown that this is not the case, bummer



    -
- learning curve extrapolation
    - based on the assumption that you have mechanisms to predict the learning curve reliably, and so extrapolation is a sensitive and valid choice. On the right there are learning curves for different architectures. Based on a few iterations and available knowledge, the method extrapolates the initial learning curves and terminates all architectures that perform poorly. The progressive neural architecture search algorithm, which is one of the approaches for neural architecture search, uses a similar method by training a surrogate model and using it to predict the performance using architectural properties

Learning Curve Extrapolation

- Requires predicting the learning curve reliably
- Extrapolates based on initial learning.
- Removes poor performers



-

- weight inheritance or network morphism
  - based on the weights of other architectures that have been trained before, similar to the way that transfer learning works. One way of achieving this is referred to as network morphism 网络射. Network morphism modifies the architecture without changing the underlying function. This is advantageous because the network inherits knowledge from the parent network, which results in methods that require only a few GPU days to design and evaluate. This allows for increasing the capacity of network successively and retaining high performance without requiring training from scratch. One advantage of this approach is that it allows for search bases that don't have an inherent upper bound on the architecture's size.

Weight Inheritance/Network Morphisms

- Initialize weights of new architectures based on previously trained architectures
  - Similar to transfer learning
- Uses **Network Morphism**
- Underlying function unchanged
  - New network inherits knowledge from parent network.
  - Computational speed up: only a few days of GPU usage
  - Network size not inherently bounded

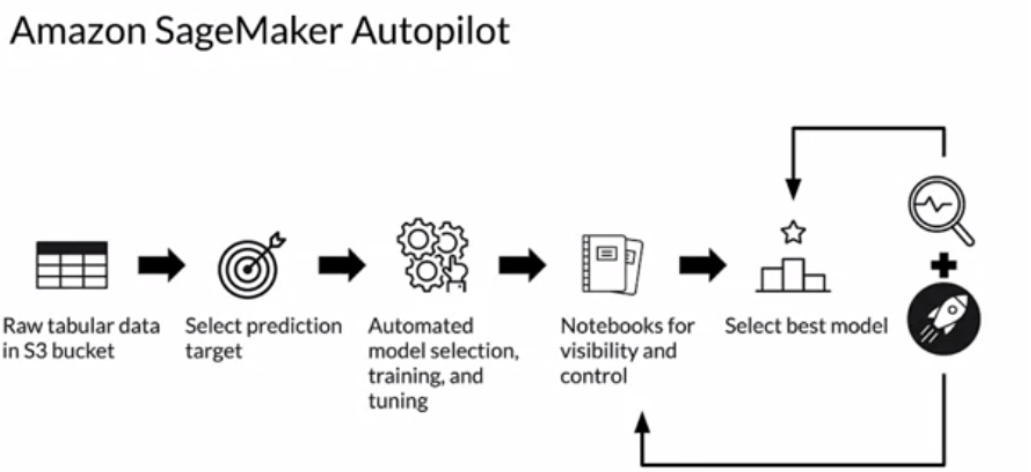
-

AutoML on the Cloud

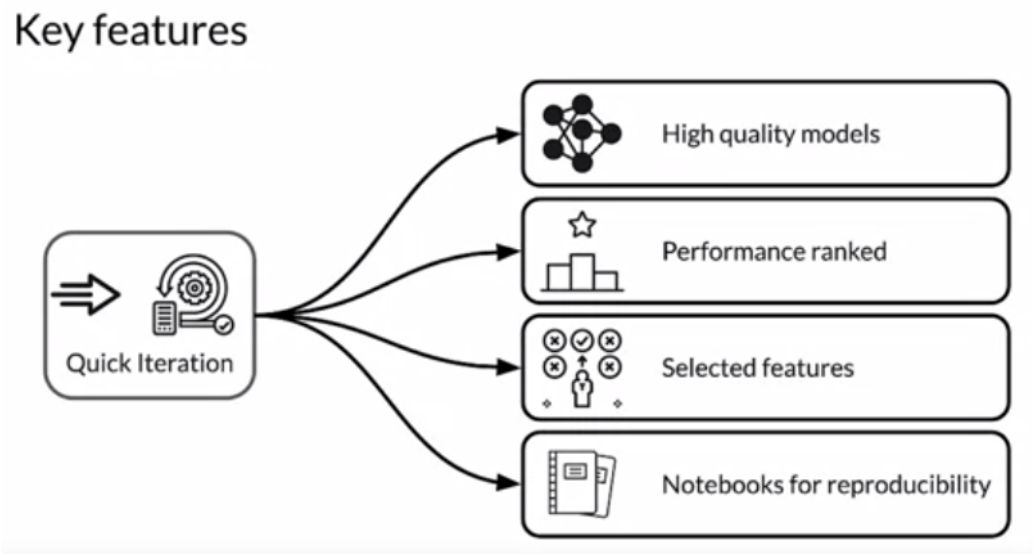
Cloud service.

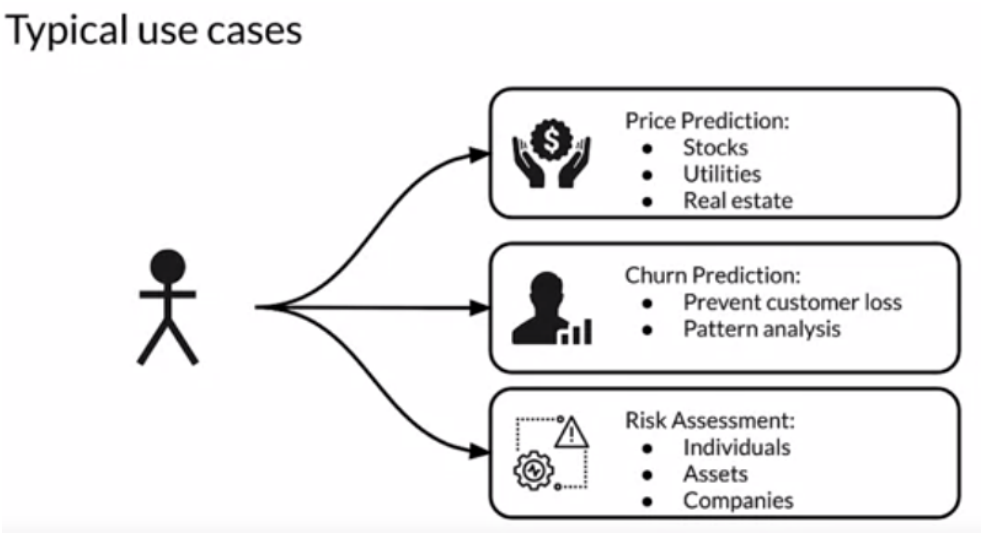
Amazon SageMaker Autopilot

- Amazon SageMaker Autopilot automatically trains and tunes the best machine learning models for classification or regression based on your data, while allowing you to maintain full control and visibility.
- Starting with your raw data, you select a label or target. Autopilot then searches for candidate models for you to review and choose from. All of these steps are documented on executable notebooks that give you full control and reproducibility of the process. This includes a leader board of model candidates to help you select the best model for your needs. You can then deploy the model to production or iterate on the recommended solutions to further improve model quality



-
- Autopilot is optimized for quick iteration. It generates **high-quality models** quickly. After the initial set of iterations, autopilot creates a **leaderboard of models ranked by performance**. You can see **which features in your dataset were selected by each model**. You can then deploy a model to production. The process of generation of the models is completely transparent. Autopilot allows you to create a SageMaker notebook from any model it created. You can then check the Notebook to dive into details of the models implementations. If need be, you can refine the model and recreate it from the Notebook at any point in time.

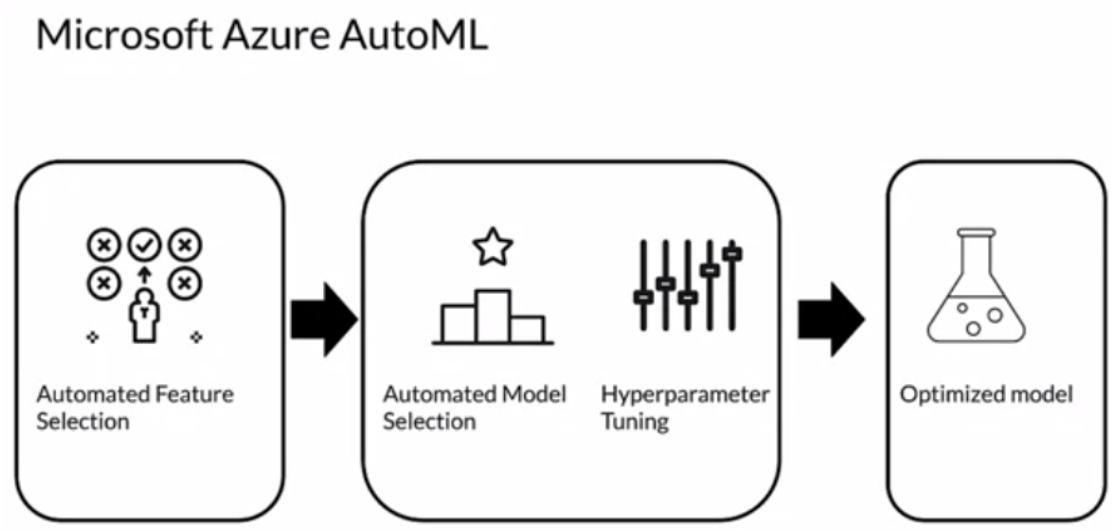
- 

- usecases:
    - it can be used to predict future prices to help you make sound investment decisions based on your historical data, such as demand, seasonal trends, and the price of other commodities. Price predictions are particularly useful in the financial services sector to predict the price of stocks or real estate, to predict real estate prices, or energy and utilities to predict the prices of natural resources
    - Churn prediction can be useful in predicting the loss of customers or churn. Companies are always looking for ways to eliminate churn. Churn prediction works by learning patterns in your existing data and identifying patterns in the new datasets so that the model can predict which customers are most likely to churn.
    - Another application is risk assessment. Risk assessment requires identifying and analyzing potential events that may negatively impact individuals, assets, and your company. Risk assessment models are trained using your existing datasets so that you can optimize the models predictions for your business.
    - 

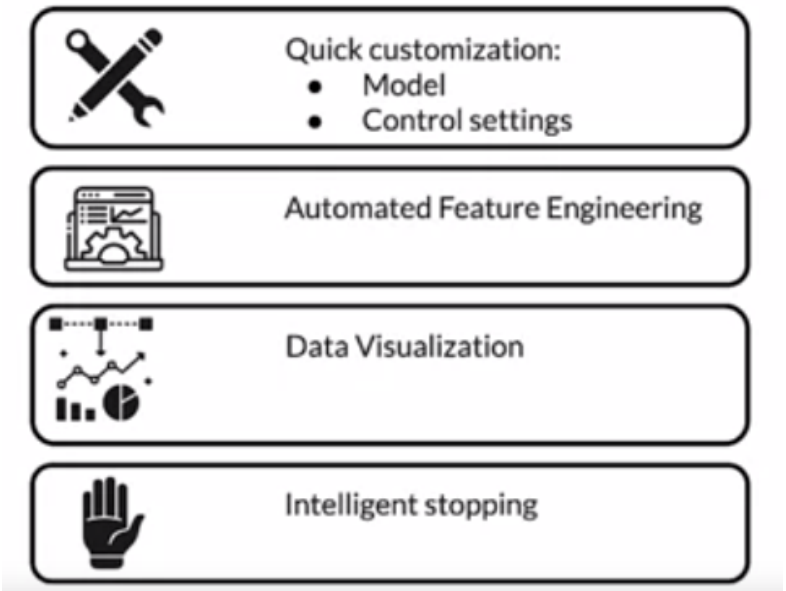
-

Microsoft Azure Automated Machine Learning

- It automates time-consuming and iterative task of model development, so basically it does AutoML. It starts with automatic feature selection, followed by model selection and hyperparameter tuning on the selected model to generate the most optimized model for the task at hand

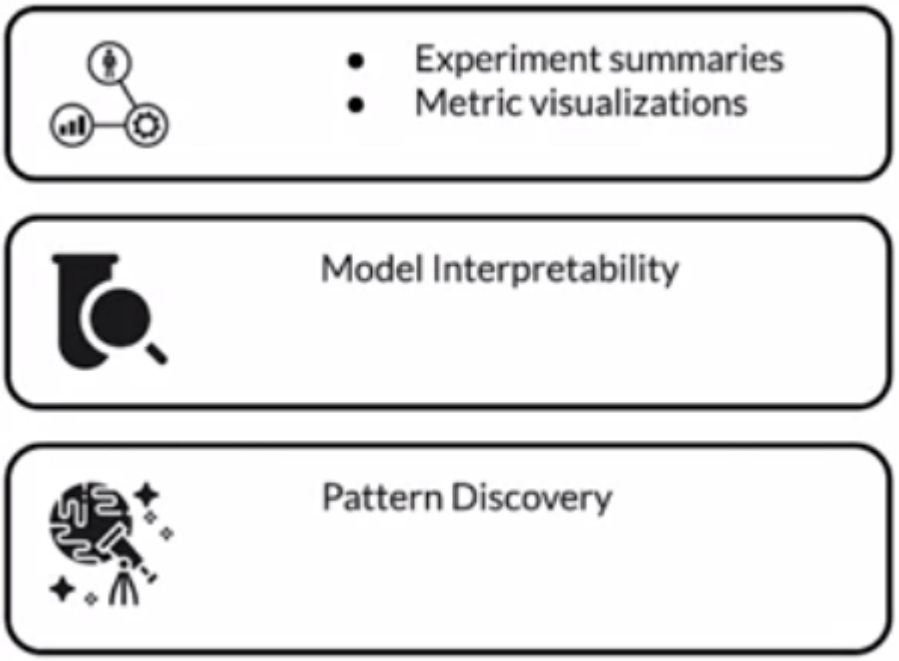


- You can either create your models using a no code UI or using code first notebooks. You can quickly customize your models, apply control settings to iterations, thresholds, validations, blocked algorithms, and other experimental criteria. It also provides tools to fully automate the feature engineering process. You can also easily visualize and profile your data to spot trends, as well as discover common errors and inconsistencies in your data. This helps you better understand the recommended actions and apply them automatically. It also provides intelligent stopping to save time on computing and prioritize the primary metric and subsampling to streamline experiment runs and speed results.



- .It also has built-in support for experiment runs summaries, and detailed metrics visualizations to help you understand your models and compare model performance. Model interpretability helps evaluate model fit for raw and engineered features and

provides insights into feature importance. You can discover patterns, perform what-if analysis, and develop deeper understanding of models to support transparency and trust in your business.

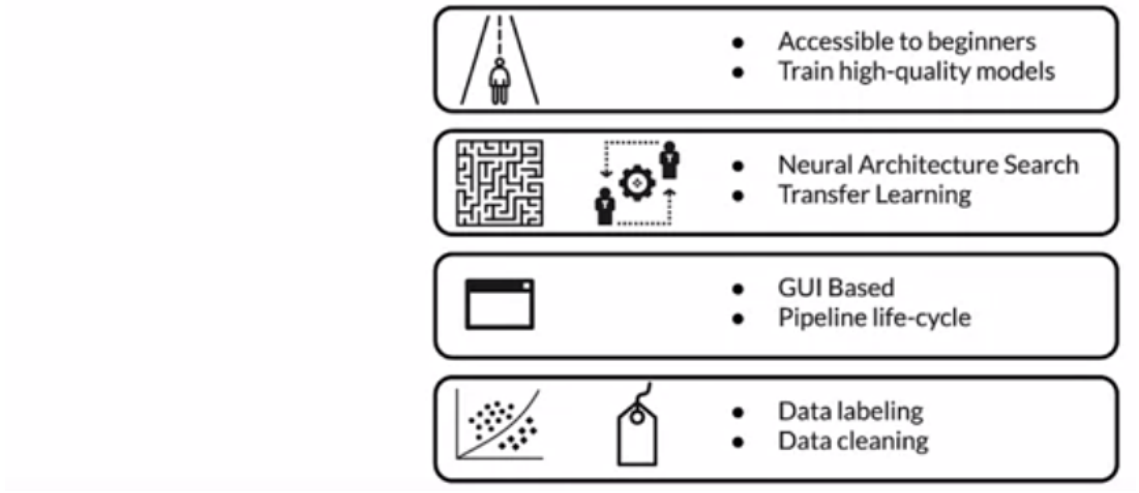


-

Google Cloud AutoML.

- a suite of machine learning products that enables developers with limited machine-learning expertise to train high-quality models specific to their business needs.
- It relies on Google, state of the art transfer learning and neural architecture search technologies. Cloud AutoML leverages more than 10 years of Google research to help your machine-learning models achieve faster performance and more accurate predictions.
- You can use Cloud AutoML, simple graphical user interface to train, evaluate, improve, and deploy models based on your data. You're really only a few minutes away from your own custom machine-learning model.
- Google's human labeling service can also put a team of people to work annotating or cleaning your labels to make sure that your models are being trained on high-quality data.

Google Cloud AutoML

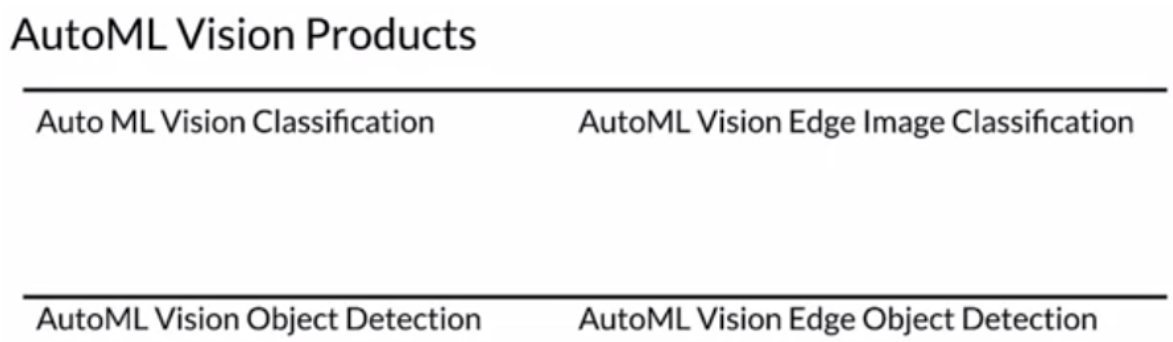


-
- Because different problems and different data need to be treated differently, Cloud AutoML isn't just one thing. It's a suite of different products, each focused towards particular use cases and data types. For example, for image data, there's **AutoML Vision** and for video data, there's **AutoML Video Intelligence**, for natural language, there's **AutoML** natural language and for translation, there's **AutoML Translation.** Finally, for general structured data, there's **AutoML Tables**.

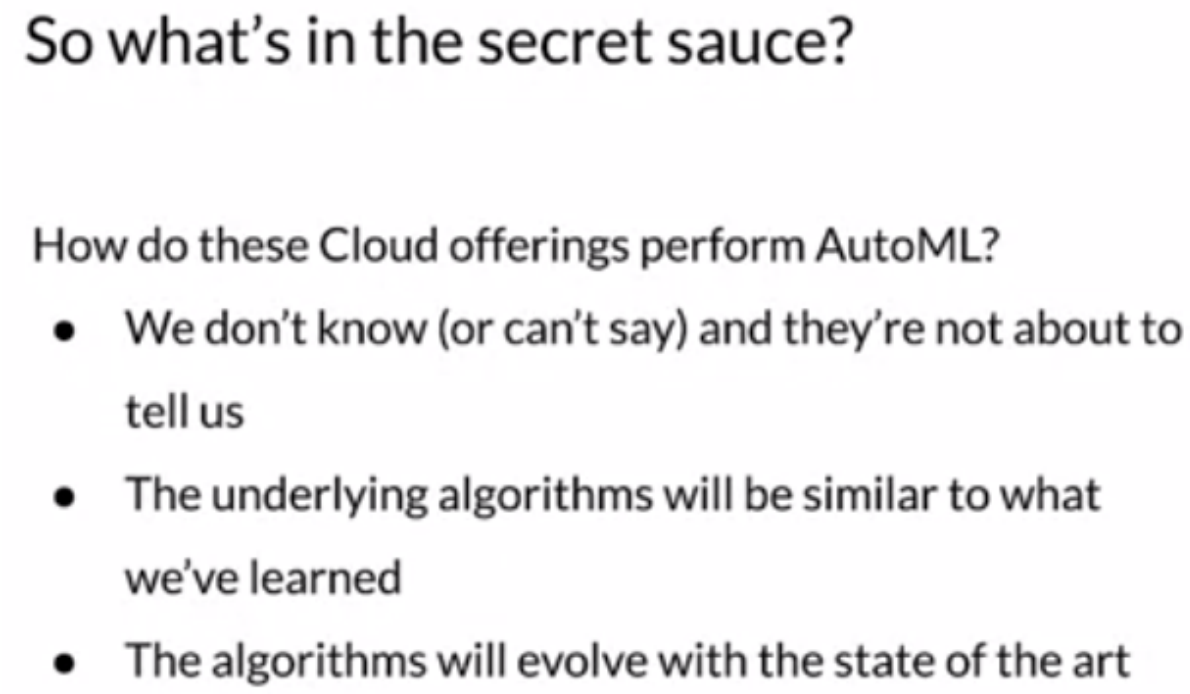
Cloud AutoML Products

Sight	Auto ML Vision Derive insights from images in the cloud or at the edge.	Auto ML Video Intelligence Enable powerful content discovery and engaging video experiences.
Language	AutoML Natural Language Reveal the structure and meaning of text through machine learning.	Auto ML Translation Dynamically detect and translate between languages.
Structured Data	AutoML Tables Automatically build and deploy state-of-the-art machine learning models on structured data.	

-
- For image data, for example, there's both vision and **classification** and Vision Object **Detection**. Then there are also Edge versions of both of these focused on optimizing for running inference at the edge in mobile applications or IoT devices.



-
- For video, there's both **Video Intelligence classification** and **video object detection**. Well, no one knows exactly or can't say if they work for a given provider. However, it is safe to assume that the algorithms at play will be similar to the ones that we've discussed in previous lessons. Also, keep in mind that ML Engineering for Production is a rapidly changing field and new technologies and developments emerge every few months. These new advancements are typically incorporated and exploited to their greatest extent by the AutoML providers.
- Also notice that these different AutoML offerings each perform the same operations that you perform or should perform as you train your model, including feature selection and feature engineering, and cleaning your labels.
- AutoML designers understand the importance of these activities.



-