
aws
INNOVATE
ONLINE CONFERENCE - AI/ML EDITION

aws INNOVATE

ONLINE CONFERENCE - AI/ML EDITION

Adding ML to your application with no ML expertise using AutoGluon

Muhyun Kim
Senior Data Scientist
Amazon ML Solutions Lab
Amazon Web Services

Common ML problems

Tabular prediction

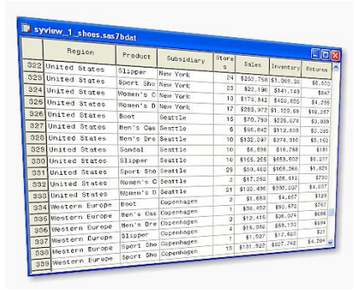
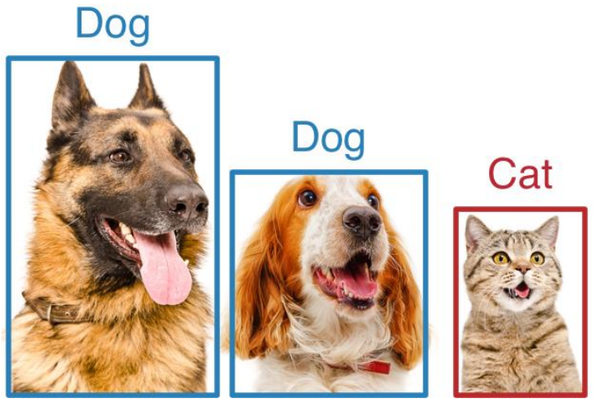


Image classification

Object detection

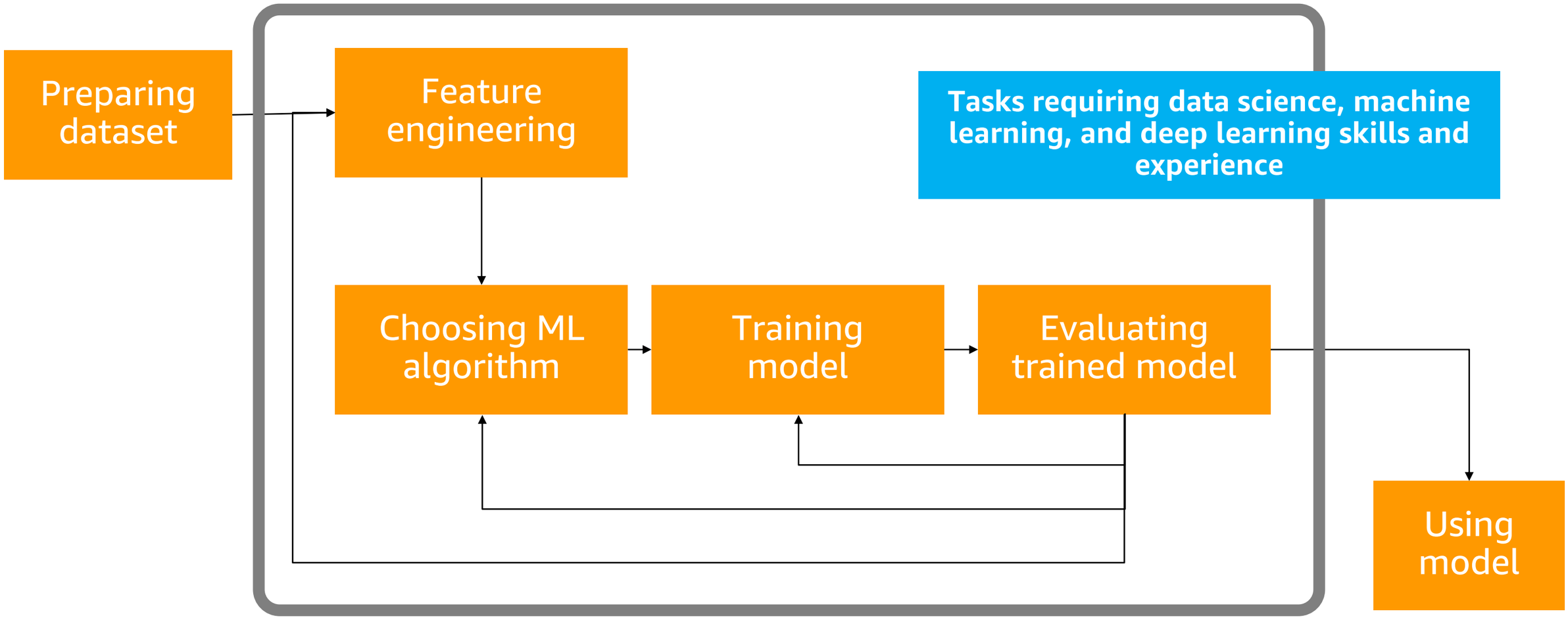


Text classification

Regression

Classification

How to solve ML problems



Required skills in building ML models

Data science for feature engineering

Machine learning and deep learning for modeling

Model tuning experience

ML and DL toolkits such as scikit-learn, TensorFlow, PyTorch, and MXNet

Do I need to learn
machine learning (ML) or **deep learning (DL)**
and an
ML/DL framework
as an application developer or data analyst?

AutoML

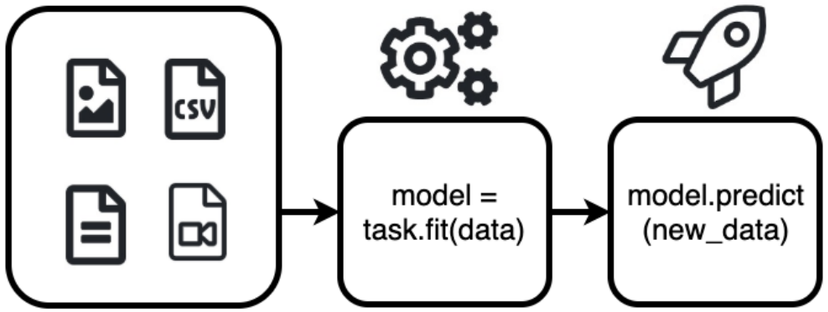
“**Automated machine learning** (**AutoML**) is the process of automating the process of applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model.” –Wikipedia

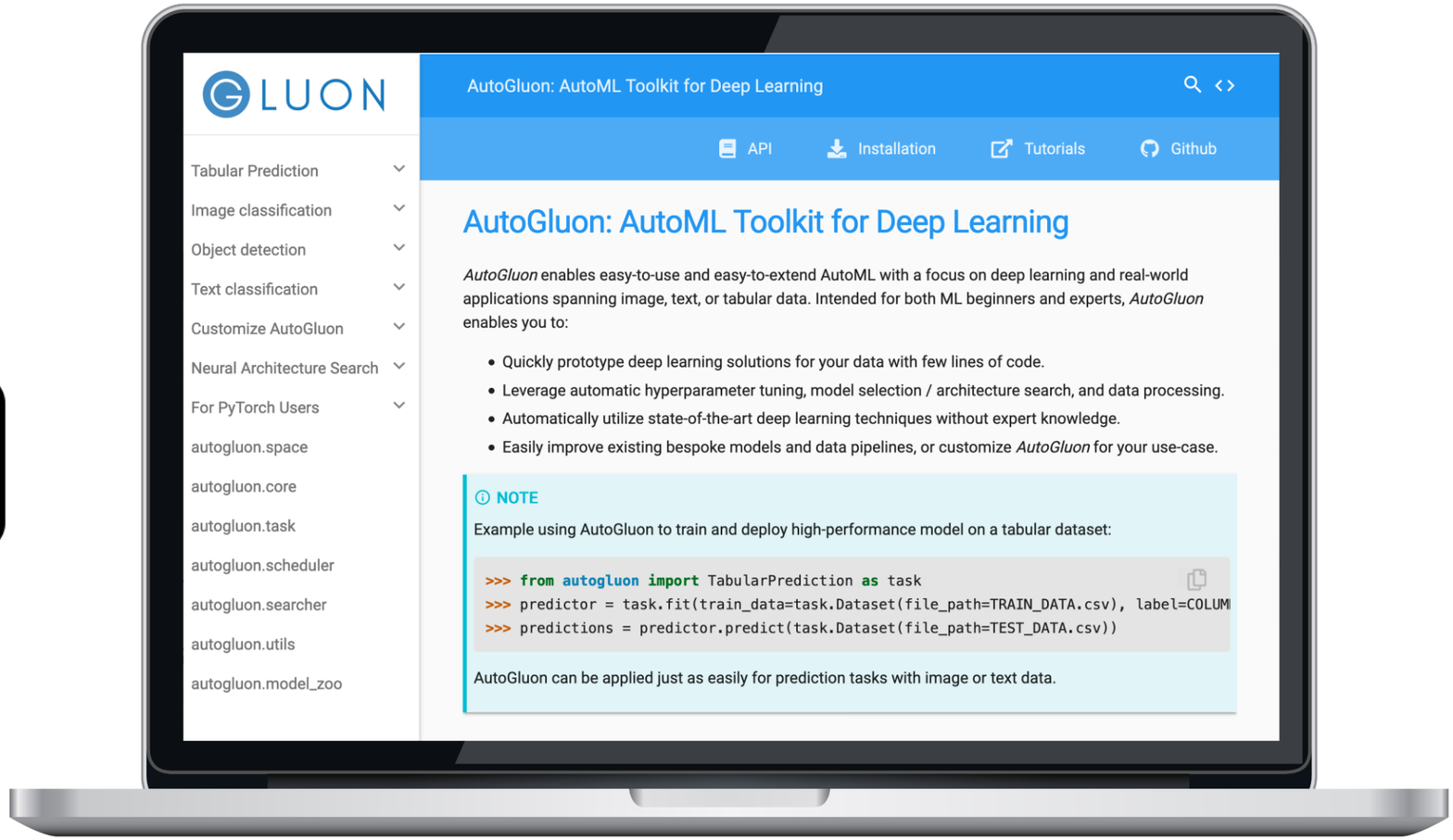
AutoML “provides methods and processes to make machine learning available for non–machine learning experts, to improve efficiency of machine learning and to accelerate research on machine learning.” –AutoML.org

- Hyperparameter optimization
- Meta-learning (learning to learn)
- Neural architecture search

AutoGluon: AutoML Toolkit for Deep Learning

https://autogluon.mxnet.io





aws

AutoGluon for “all”

Developers/analysts with no ML skill

Automating all ML pipelines
- Feature engineering
- Model selection
- Model training
- Hyperparameter optimization

ML experts

- Quick model prototyping for baseline
- Hyperparameter optimization
- Optimizing custom models

Researchers

- Model optimization
- Searching for new architectures

What you can do with AutoGluon

- Quick prototyping achieving state-of-the-art performance for the following:
  - Tabular prediction
  - Image classification
  - Object detection
  - Text classification
- Customizing model searching
- Hyperparameter optimization on model training in Python or PyTorch
- Neural architecture searching

Basic usage of AutoGluon

aws INNOVATE
ONLINE CONFERENCE - AI/ML EDITION

3 simple steps to get the best ML model

Step 1. Prepare your dataset

Step 2. Load the dataset for training ML

Step 3. Call `fit()` to get the best ML model

What happens behind the scenes

Loading the dataset for training ML

- ML problem defined (binary/multiple classification or regression)
- Feature engineering for each model being trained
- Missing value handling
- Splitting dataset into training and validation

Calling `fit()` to get the best ML model

- Training models
- Hyperparameter optimization
- Model selection

Common ML problems

Tabular prediction

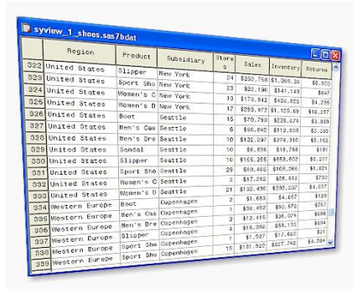
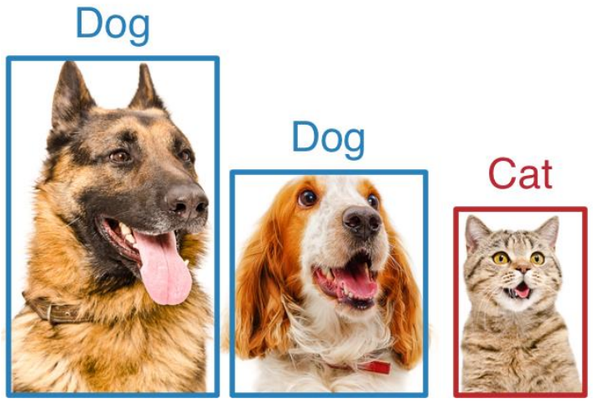


Image classification

Object detection



Text classification

Regression

Classification

ML algorithms for tabular prediction

- Random Forest
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- XT (extremely randomized trees)
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
- K-nearest neighbors
  - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- CatBoost: Gradient boosting on decision trees
  - https://catboost.ai/
- LightGBM
  - https://lightgbm.readthedocs.io
- Neural network

Let's prepare a tabular dataset

A structured dataset stored in CSV format where:

- Each row represents an example
- Each column represents the measurements of some variable or feature

Files stored in either an Amazon S3 bucket or the local file system

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
1	25	Private	178478	Bachelors	13	Never-married	Tech-support	Own-child	White	Female	0	0	40	United-States	<=50K
2	23	State-gov	61743	5th-6th	3	Never-married	Transport-moving	Not-in-family	White	Male	0	0	35	United-States	<=50K
3	46	Private	376789	HS-grad	9	Never-married	Other-service	Not-in-family	White	Male	0	0	15	United-States	<=50K
4	55	?	200235	HS-grad	9	Married-civ-spouse	?	Husband	White	Male	0	0	50	United-States	>50K
5	36	Private	224541	7th-8th	4	Married-civ-spouse	Handlers-cleaners	Husband	White	Male	0	0	40	El-Salvador	<=50K
6	51	Private	178054	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	40	?	>50K
7	33	Private	263561	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	60	United-States	>50K
8	46	Private	173613	HS-grad	9	Divorced	Adm-clerical	Not-in-family	White	Female	0	0	40	United-States	<=50K
9	18	Private	214617	Some-college	10	Never-married	Handlers-cleaners	Own-child	White	Male	0	0	30	United-States	<=50K
10	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales	Husband	White	Male	0	0	45	United-States	<=50K
11	41	Private	225892	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	48	United-States	>50K

Some data found in **Adult Data Set** (https://archive.ics.uci.edu/ml/datasets/adult)

aws

Step 0: Install AutoGluon

```
# CUDA 10.0 and a GPU for object detection is recommended
# We install MXNet to utilize deep learning models

# For Linux with GPU installed
pip install --upgrade mxnet-cu100

# For Linux without GPU
pip install --upgrade mxnet

# Install AutoGluon package
pip install autogluon
```

Step 1: Loading dataset

```
from autogluon import TabularPrediction as task


train_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/train_data.csv'
train_data = task.Dataset(file_path=train_path)
```

file_path : str (optional)
Path to the data file (may be on local filesystem or URL to cloud s3 bucket).

df : `pandas.DataFrame` (optional)
If you already have your data in a pandas Dataframe, you can directly provide it by specifying `df`.

feature_types : dict (optional)
Mapping from column_names to string describing data type of each column. If not specified, AutoGluon's fit() will automatically infer what type of data each feature contains.

subsample : int (optional)
f specified = k, we only keep first k rows of the provided dataset.

name : str (optional)
Optional name to assign to dataset

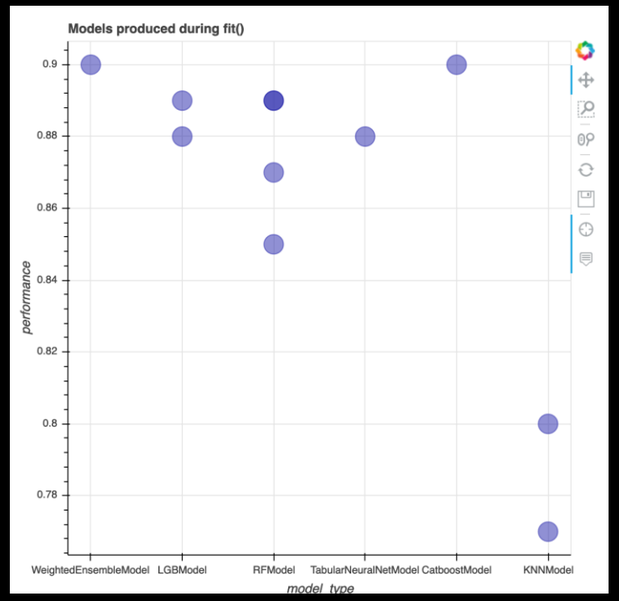
Step 2: Training ML models

```
predictor = task.fit(train_data, label='class', output_directory='ag-example-out/')
```

dataset to be used for ML training

column name to be predicted

directory where the trained models are saved




```
ag-example-out/
├── SummaryOfModels.html
├── learner.pkl
└── models
    ├── CatboostClassifier
    │   └── model.pkl
    ├── ExtraTreesClassifierEntr
    ├── ExtraTreesClassifierGini
    ├── KNeighborsClassifierDist
    ├── KNeighborsClassifierUnif
    ├── LightGBMClassifier
    ├── LightGBMClassifierCustom
    ├── NeuralNetClassifier
    │   ├── net.params
    │   ├── tabularNN.pkl
    │   └── temp_net.params
    ├── RandomForestClassifierEntr
    ├── RandomForestClassifierGini
    ├── trainer.pkl
    └── weighted_ensemble_l1
```


aws

Parameters of *fit()*

https://autogluon.mxnet.io/api/autogluon.task.html#autogluon.task.TabularPrediction.fit

```
def fit(train_data, label, tuning_data=None, output_directory=None, problem_type=None, eval_metric=None,
        hyperparameter_tune=False, feature_prune=False, auto_stack=False, holdout_frac=None,
        num_bagging_folds=0, num_bagging_sets=None, stack_ensemble_levels=0,
        hyperparameters = {
                            'NN': {'num_epochs': 500},
                            'GBM': {'num_boost_round': 10000},
                            'CAT': {'iterations': 10000},
                            'RF': {'n_estimators': 300},
                            'XT': {'n_estimators': 300},
                            'KNN': {},
                            'custom': ['GBM'],
                           },
        enable_fit_continuation=False,
        time_limits=None, num_trials=None, search_strategy='random', search_options={},
        nthreads_per_trial=None, ngpus_per_trial=None, dist_ip_addrs=[], visualizer='none',
        verbosity=2, **kwargs):
```

‘random’ (random search), ‘skopt’ (SKopt Bayesian optimization), ‘grid’ (grid search), ‘hyperband’ (Hyperband), ‘rl’ (reinforcement learner)

‘mxboard’, ‘tensorboard’, ‘none’

aws

Step 3: Evaluate the model

```
test_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/test_data.csv'
test_data = task.Dataset(file_path=test_path)
leaderboard = predictor.leaderboard(test_data)
```

	model	score_test	score_val	fit_time	pred_time	stack_level
0	weighted_ensemble_l1	0.876958	0.8748	1.799241	0.001847	1
1	CatboostClassifier	0.876548	0.8740	39.542085	0.039164	0
2	LightGBMClassifier	0.875729	0.8672	7.899164	0.033295	0
3	LightGBMClassifierCustom	0.874092	0.8676	34.620379	0.046849	0
4	RandomForestClassifierEntr	0.860375	0.8504	10.123323	0.226658	0
5	RandomForestClassifierGini	0.859044	0.8504	10.425733	0.236528	0
6	NeuralNetClassifier	0.858327	0.8544	265.441416	0.479654	0
7	ExtraTreesClassifierGini	0.846863	0.8388	7.431412	0.232303	0
8	ExtraTreesClassifierEntr	0.845122	0.8408	7.236258	0.222955	0
9	KNeighborsClassifierUnif	0.774695	0.7736	0.324976	0.120106	0
10	KNeighborsClassifierDist	0.762105	0.7644	0.319923	0.111614	0

evaluation metric specified in fit()

Step 4: Use the model in your app

```
predictor = task.load('ag-example-out/')
```

loading models saved in the specified directory

```
test_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/test_data.csv'
test_data = task.Dataset(file_path=test_path)
y_test = test_data['class']
test_data_nolabel = test_data.drop(labels=['class'],axis=1)
```

```
y_pred = predictor.predict(test_data_nolabel)
```

predicting on the given dataset using the best model

predicted values in numpy array by default

aws

Parameters of *predict()*

https://autogluon.mxnet.io/api/autogluon.task.html#autogluon.task.tabular_prediction.TabularPredictor.predict

dataset : `TabularDataset` or `pandas.DataFrame`
The dataset to make predictions for. Should contain same column names as training Dataset and follow same format.

model : str (optional)
The name of the model to get predictions from. Defaults to None, which uses the highest scoring model on the validation set.

as_pandas : bool (optional)
Whether to return the output as a pandas Series (True) or numpy array (False).

use_pred_cache : bool (optional)
Whether to used previously-cached predictions for table rows we have already predicted on before.

add_to_pred_cache : bool (optional)
Whether these predictions should be cached for reuse in future `predict()` calls on the same table rows.

Demo

aws INNOVATE
ONLINE CONFERENCE - AI/ML EDITION

Image classification

```
from autogluon import ImageClassification as task

# Loading dataset
dataset = task.Dataset('./data/shopeeiet/train')

# Train image classification models
time_limits = 10 * 60 # 10mins
classifier = task.fit(dataset,
                      time_limits=time_limits,
                      ngpus_per_trial=1)

# Test the trained model
test_dataset = task.Dataset('./data/shopeeiet/test')
inds, probs, probs_all = classifier.predict(test_dataset)
```

```
shopeeiet/train
    ├── BabyBibs
    ├── BabyHat
    ├── BabyPants
    ├── ...
shopeeiet/test
    ├── ...
```


aws

For advanced features of AutoGluon

Other AutoML tasks

- Image classification
- Object detection
- Text classification

Hyperparameter optimization

- Hyperparameter search space and search algorithm customization
- Distributed search

Neural architecture search

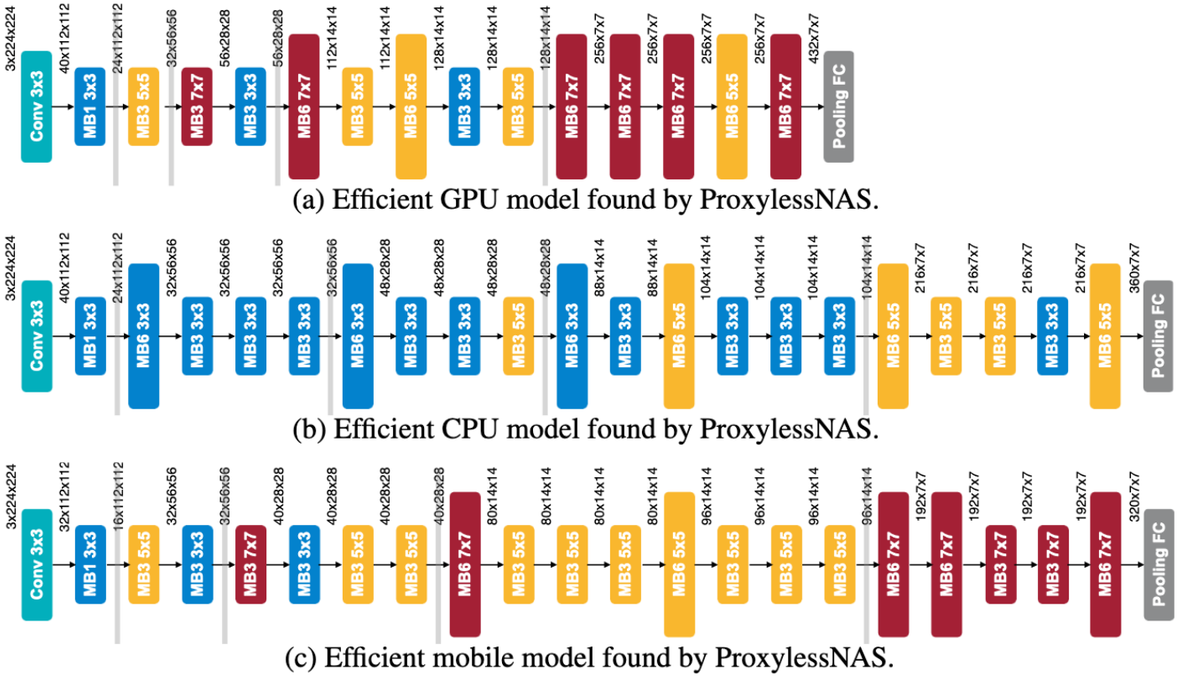
- ENAS/ProxylessNAS

Efficient NAS on target hardware: ProxylessNAS

https://autogluon.mxnet.io/tutorials/nas/enas_mnist.html

Model	Top-1	Top-5	Mobile Latency	Hardware -aware	No Proxy	No Repeat	Search cost (GPU hours)
MobileNetV1 [16]	70.6	89.5	113ms	-	-	✗	Manual
MobileNetV2 [30]	72.0	91.0	75ms	-	-	✗	Manual
NASNet-A [38]	74.0	91.3	183ms	✗	✗	✗	48,000
AmoebaNet-A [29]	74.5	92.0	190ms	✗	✗	✗	75,600
MnasNet [31]	74.0	91.8	76ms	✓	✗	✗	40,000
MnasNet (our impl.)	74.0	91.8	79ms	✓	✗	✗	40,000
Proxyless-G (mobile)	71.8	90.3	83ms	✗	✓	✓	200
Proxyless-G + LL	74.2	91.7	79ms	✓	✓	✓	200
Proxyless-R (mobile)	**74.6**	**92.2**	78ms	✓	✓	✓	200

ProxylessNAS achieves state-of-the art accuracy (%) on ImageNet (under mobile latency constraint $\leq 80ms$) with 200× less search cost in GPU hours. "LL" indicates latency regularization loss.



(a) Efficient GPU model found by ProxylessNAS.

(b) Efficient CPU model found by ProxylessNAS.

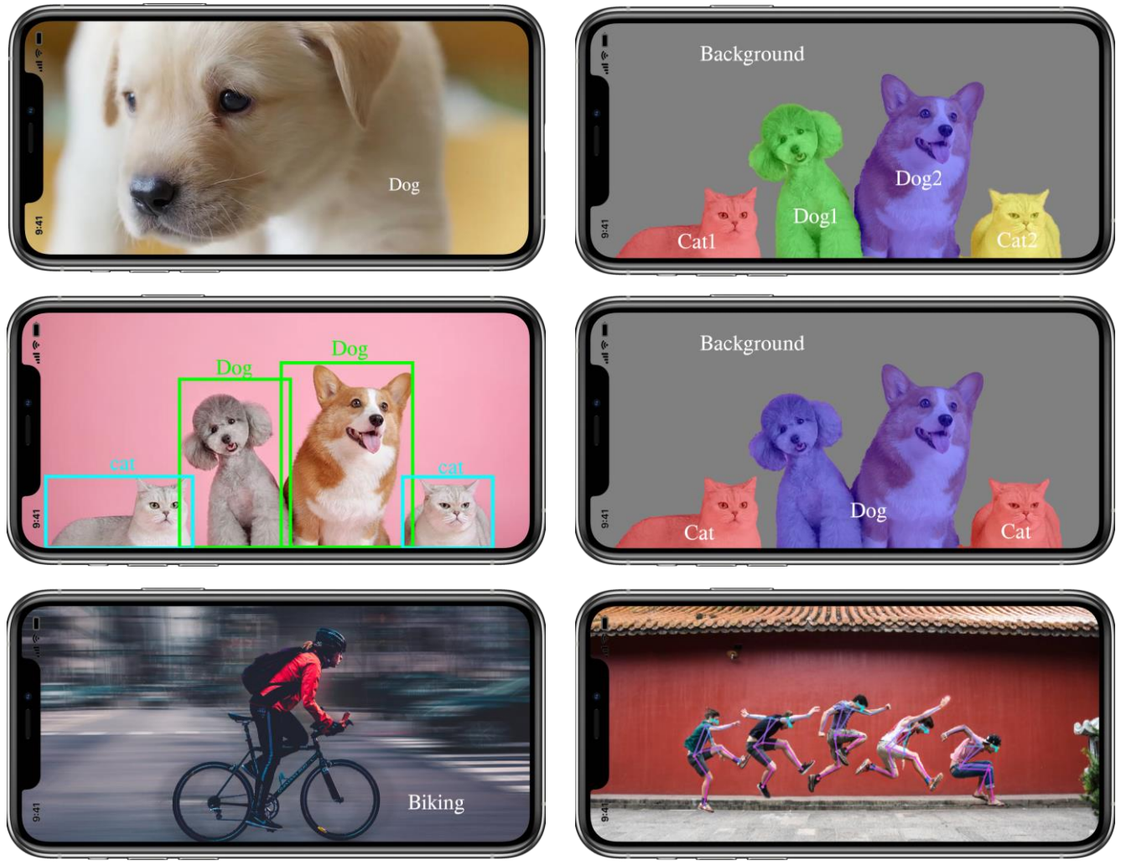
(c) Efficient mobile model found by ProxylessNAS.

(Source) "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware (last revised Feb. 23, 2019)."

aws

More toolkits for developers

GluonCV



Deep Java Library

Open source library to build and deploy Deep Learning in Java

Framework Agnostic

Built for Java developers

Ease of deployment

```
        model.load(modelDir,  modelName: "mlp");

        Translator<BufferedImage, Classifications> translator = new MyTranslator();
        try (Predictor<BufferedImage, Classifications> predictor =
                model.newPredictor(translator)) {
            return predictor.predict(img);
        }
    }
}

private static final class MyTranslator implements Translator<BufferedImage, Classifications> {

    private List<String> classes;

    public MyTranslator() {
        classes = IntStream.range(0, 10).mapToObj(String::valueOf).collect(Collectors.toList());
    }
```


aws

Resources

AutoGluon (https://autogluon.mxnet.io)

GluonCV (https://gluon-cv.mxnet.io)

AWS Computer Vision: Getting Started with GluonCV (https://www.coursera.org/learn/aws-computer-vision-gluoncv)

Deep Java Library (https://djl.ai)

Dive into Deep Learning (https://d2l.ai, https://ko.d2l.ai)

Machine Learning on AWS (https://ml.aws)

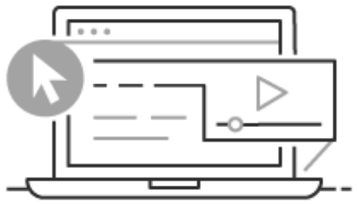
Amazon ML Solutions Lab (https://aws.amazon.com/ml-solutions-lab)

AWS 머신러닝(ML) 교육 및 자격증

Amazon의 개발자와 데이터 과학자를 교육하는 데 직접 활용 되었던 커리큘럼을 기반으로 학습하세요!

전체 팀을 위한 머신러닝 교육

비즈니스 의사 결정자, 데이터 과학자, 개발자, 데이터 플랫폼 엔지니어 등 역할에 따라 제공되는 맞춤형 학습 경로를 확인하세요.



원하는 방법으로! 교육 유연성 제공

약 65개 이상의 온라인 과정 및 AWS 전문 강사를 통해 실습과 실적용의 기회가 제공되는 강의실 교육이 준비되어 있습니다.

전문성에 대한 검증

업계에서 인정받는 'AWS 공인 머신러닝 – 전문분야' 자격증을 통해 머신러닝 모델을 구축, 학습, 튜닝 및 배포하는 데 필요한 전문 지식이 있음을 인증할 수 있습니다.

https://aws.amazon.com/ko/training/learning-paths/machine-learning/

AWS Innovate 온라인 컨퍼런스에 참석해주셔서 대단히 감사합니다.

저희가 준비한 내용, 어떻게 보셨나요?
더 나은 세미나를 위하여 **설문을 꼭 작성해 주시기 바랍니다.**


aws-korea-marketing@amazon.com


twitter.com/AWSKorea


facebook.com/amazonwebservices.ko


youtube.com/user/AWSKorea


slideshare.net/awskorea


twitch.tv/aws

Thank you!

aws INNOVATE
ONLINE CONFERENCE - AI/ML EDITION