AWS

MACHINE LEARNING

CERTIFICATION

DOMAIN #3: MODELING (36% EXAM)

AWS ML CERTIFICATION EXAM DOMAINS

Domain	% of Examination
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: Machine Learning Implementation and Operations	20%
TOTAL	100%

Source: https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty_Exam%20Guide%20(1).pdf

DOMAIN #3 OVERVIEW:

SECTION #8: MACHINE AND DEEP LEARNING BASICS – PART #1

- Artificial Neural Networks Basics: Single Neuron Model
- Activation Functions
- Multi-Layer Perceptron Model
- How do Artificial Neural Networks Train?
- ANN Parameters Tuning – Learning rate and batch size
- Tensorflow playground
- Gradient Descent and Backpropagation
- Overfitting and Under fitting
- How to overcome overfitting?
- Bias Variance Trade-off
- L1 Regularization
- L2 Regularization

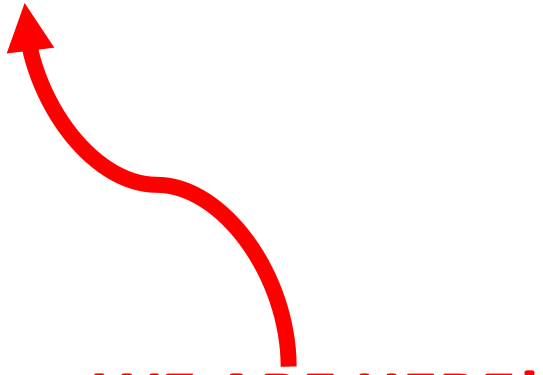
SECTION #9: MACHINE AND DEEP LEARNING BASICS – PART #2

- Artificial Neural Networks Architectures
- Convolutional Neural Networks
- Recurrent Neural Networks
- Vanishing Gradient Problem
- LSTM Networks
- Model Performance Assessment – Confusion Matrix
- Model Performance Assessment – Precision, recall, F1-score
- Model Performance Assessment – ROC, AUC, Heatmap, and RMSE
- K-Fold Cross validation
- Transfer Learning
- Ensemble Learning – Bagging and Boosting

DOMAIN #3 OVERVIEW:

SECTION #10: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #1

- AWS SageMaker
- Deep Learning on AWS
- SageMaker Built-in algorithms
- Object Detection
- Image Classification
- Semantic Segmentation
- SageMaker Linear Learner
- Factorization Machines
- XG-Boost
- SageMaker Seq2Seq
- SageMaker DeepAR
- SageMaker Blazing Text



WE ARE HERE!

SECTION #11: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #2

- Object2Vec
- Random Cut Forest
- Neural Topic Model
- LDA
- K-Nearest Neighbours (KNN)
- K Means
- Principal Component Analysis (PCA)
- IP insights
- Reinforcement Learning
- Automatic Model Tuning
- SageMaker and Spark

DOMAIN #3 OVERVIEW:

SECTION #12: MACHINE AND DEEP LEARNING IN AWS – HIGH LEVEL AI/ML PART #3

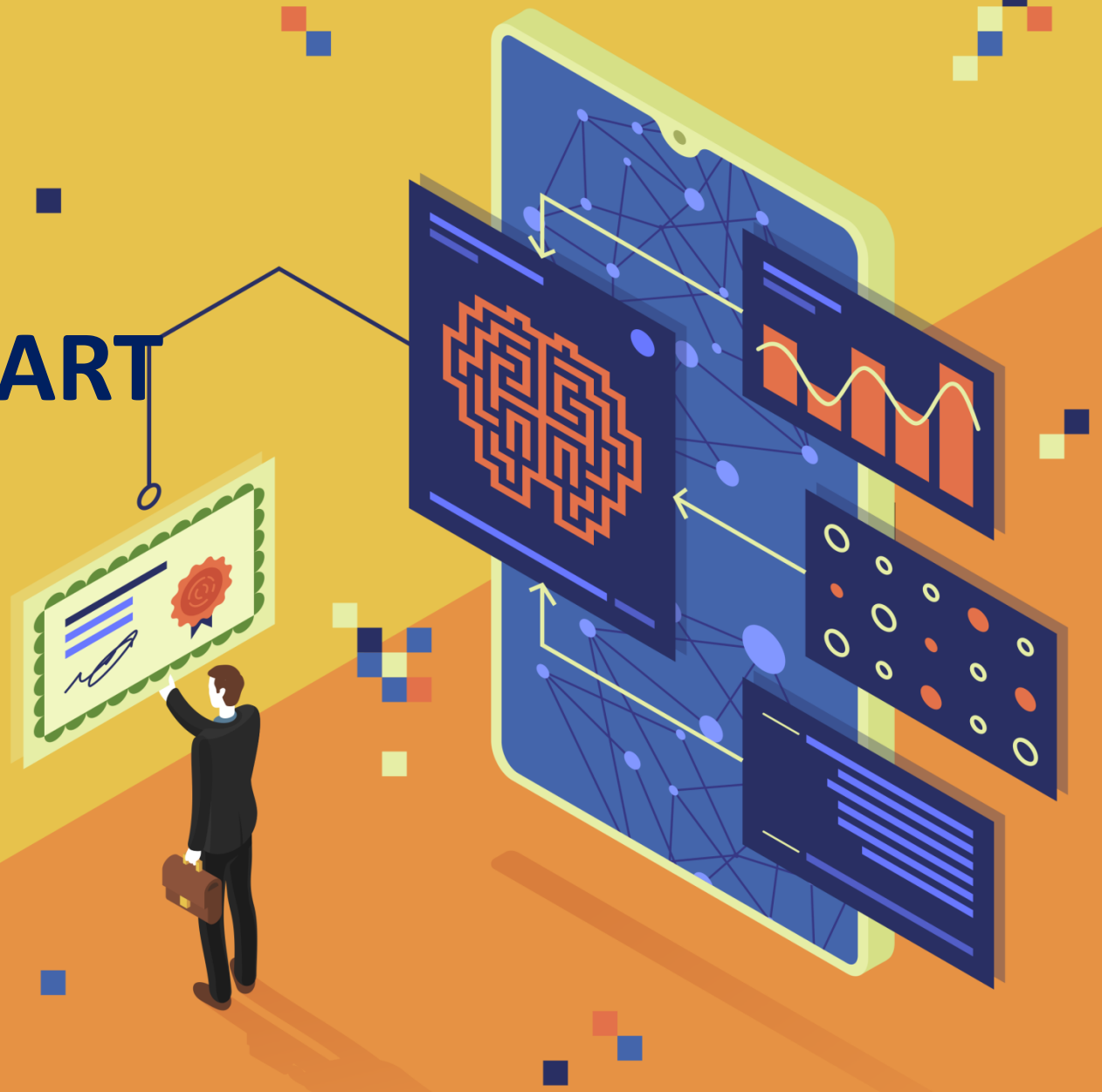
- ReKognition
- Amazon Comprehend and Comprehend Medical
- Translate
- Transcribe
- Polly
- Forecast
- Lex
- Personalize
- Textract
- AWS DeepLens
- AWS DeepRacer

RECALL OUR MINI CHALLENGE AND PRIZE!

- For those of you who will successfully complete the entire section and watch the videos till the end, they will receive a valuable prize!

https://pixabay.com/illustrations/trophy-win-prize-transparent-1414791/

AWS SAGEMAKER – PART #1



AMAZON SAGEMAKER

- Amazon SageMaker is a fully-managed machine learning workflow platform that provides services on data labeling, model building, training, tuning and deployment.
- SageMaker allows data scientists and developers to build scalable AI/ML models easily and efficiently.
- Models could be deployed in production at a much faster rate and with a fraction of the cost.
- Let's explore SageMaker: https://aws.amazon.com/sagemaker/#

BUILD

- SageMaker offers data labeling service
- Prebuilt available notebooks with state of the art algorithms on AWS marketplace

TRAIN

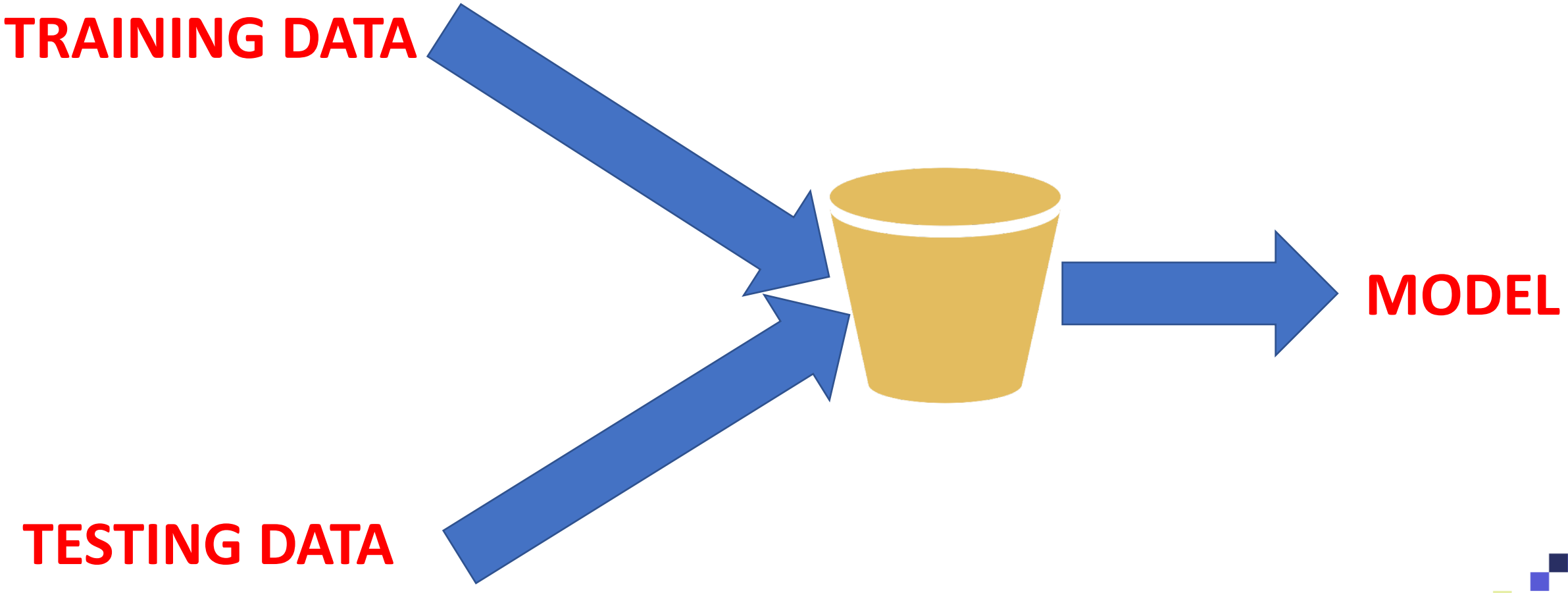
- Train models using EC2 instances (on-demand and spot)
- Manage environments for training
- Hyperparameters optimization for model tuning

DEPLOY

- Easily deploy and scale models
- Autoscaling with 75% savings

AMAZON SAGEMAKER SERVICES

TRAINING/TESTING DATA IN S3



PIPE MODE VS. FILE MODE

- You can run Amazon SageMaker built in algorithms in either:
  - File Mode (csv, JSON, Parquet)
  - Pipe Mode (recordIO-protobuf)
- Pipe mode allows for much faster training and less disk space since data is streamed directly to the training instance container instead of being downloaded first. Data is obtained from Amazon S3 using an optimized multi-thread process.
- For optimal performance, recordIO-protobuf format in pipe mode is recommended.

GPU Vs. CPU Vs. FPGA Vs. ASIC

	CPU	FPGA	GPU	ASIC
Acronym?	Center Processing Unit (CPU)	Field-Programmable Gate Array (FPGA)	Graphics Processing Unit (GPU)	Application Specific Integrated Circuit (ASIC)
What is it?	CPU is a basic sequential processor	FPGA is an integrated circuit designed to be configurable after manufacturing.	GPU is designed to process images and graphics	ASIC is a circuit designed for a specific target application.
Advantages?	• Easy to program • Generalist that could be applied in many applications. • Cheap	• Could be configured after being installed in the field. • Supports parallel processing	• Geared towards image analysis and graphics applications	• Optimized for a specific application with ultimate performance and power consumption
Disadvantages?	• Slow with no or limited parallel processing.	• Hard to program	• High power consumption	• Rigid • High cost • Long development time

AWS SAGEMAKER – PART #2

AMAZON SAGEMAKER COMPONENTS

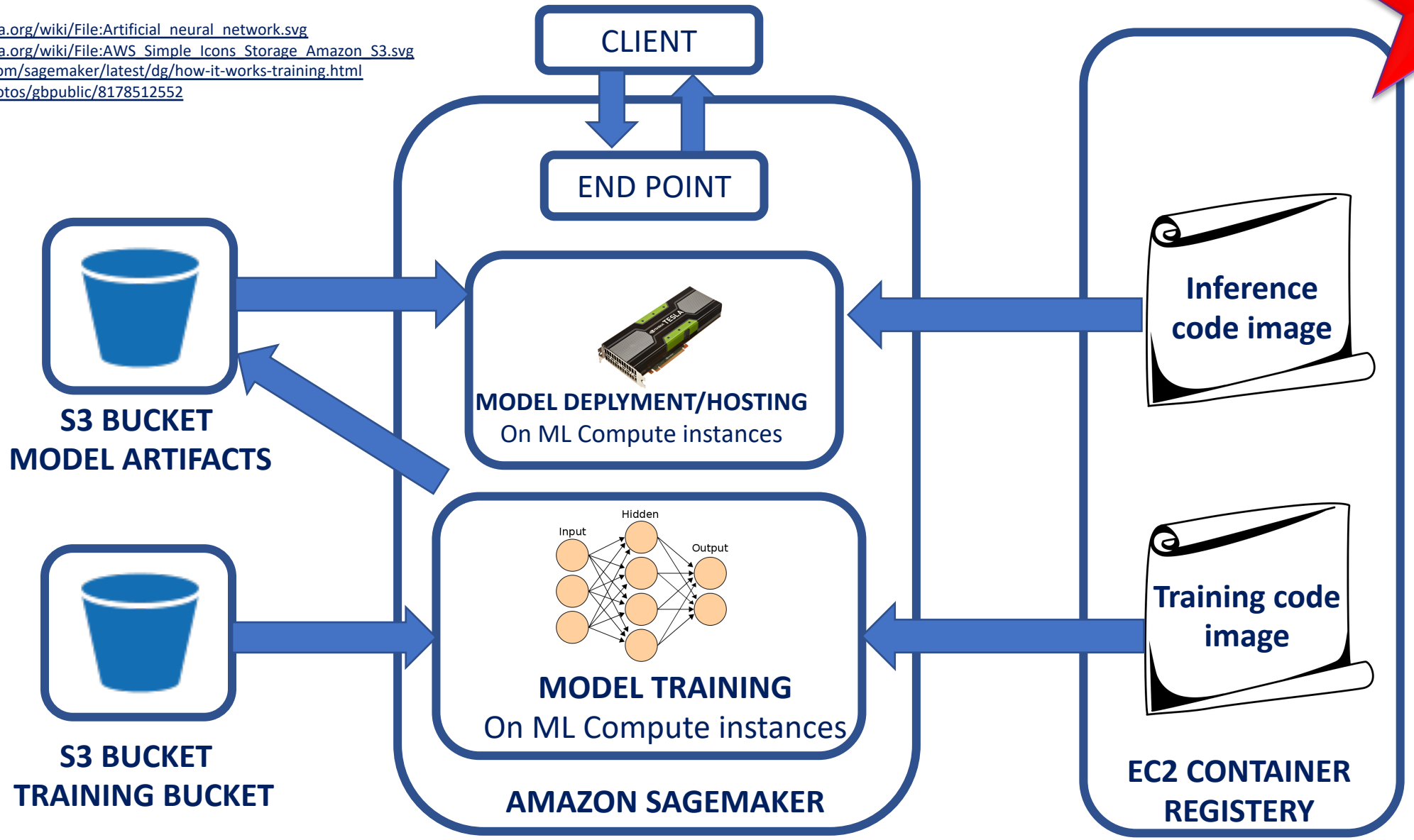
- Two components are present in Amazon SageMaker:
  - Model training
  - Model deployment.
- To start training an AI/ML model using Amazon SageMaker, you will need to create a training job with the following:
  - **Amazon S3 bucket URL (training data):** where the training data is located.
  - **Compute resources:** Amazon SageMaker will train the model using instances managed by Amazon SageMaker.
  - **Amazon S3 bucket URL (Output):** this bucket will host the output from the training.
  - **Amazon Elastic Container Registry path:** where the training code is stored.
- Amazon SageMaker launches an ML compute instances once a training job is initiated.
- Amazon SageMaker uses: (1) training code and (2) training dataset to train the model.
- Amazon SageMaker saves the trained model artifacts in an S3 bucket.

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

AMAZON SAGEMAKER MODEL TRAINING AND DEPLOYMENT OVERVIEW

Source:
https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg
https://commons.wikimedia.org/wiki/File:AWS_Simple_Icons_Storage_Amazon_S3.svg
https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html
https://www.flickr.com/photos/gbpublic/8178512552



TRAINING OPTIONS OFFERED BY SAGEMAKER

USE AN ALGORITHM PROVIDED BY AMAZON SAGEMAKER

- Amazon SageMaker provides ready, off the shelve training algorithms such as: Linear Learner Algorithm and the XGBoost Algorithm, K Means, Principal Component Analysis, image classification, LDA, Sequence to Sequence Algorithm.

USE APACHE SPARK WITH AMAZON SAGEMAKER

- Apache Spark can be used to train models with Amazon SageMaker.

CUSTOM CODE TRAINING USING POPULAR DEEP LEARNING FRAMEWORKS

- custom python code with TensorFlow or Apache MXNet for model training.

USE YOUR OWN CUSTOM ALGORITHMS:

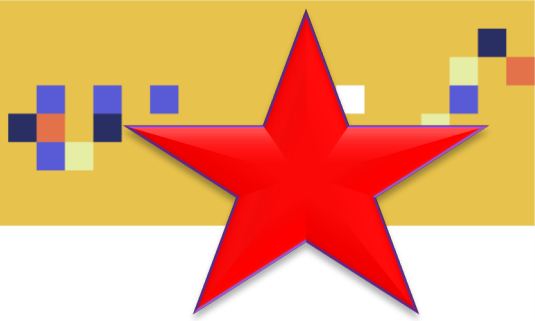
- The code could be placed in a docker contained and then and registry path of the image could be provided in an Amazon SageMaker CreateTrainingJob API call.

AWS MARKETPLACE

- choose an algorithm from Amazon marketplace, https://aws.amazon.com/marketplace/solutions/machine-learning

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

MODEL DEPLOYMENT BY SAGEMAKER



- After the AI/ML model is trained, it can be deployed as follows:
  - **For Single Prediction at a time:** set up a persistent endpoint using Amazon SageMaker hosting services.
  - **For Multiple Predictions:** Use Amazon SageMaker batch transform in order to obtain predictions for an entire dataset.
- **SageMaker Inference Pipeline:** SageMaker offers tools to process batch transforms in a pipeline format.
- **Batch Transform:** is used to allow preprocessing of large datasets and performing model inferencing quickly/efficiently without a need to have a persistent endpoint.
- **SageMaker Automatically Scaling:** as the workload changes throughout the day, SageMaker can dynamically adjust the number of instances provisioned so it could save money and compute resources.
- **Amazon SageMaker Elastic Inference (EI):** EI could be used to speed up inference and reduce latency by adding an accelerator without the need of having a dedicated GPU (which will cost much more)
- **Amazon SageMaker Neo:** Used to train TensorFlow, Apache MXNet, PyTorch, ONNX, and XGBoost models once and optimize them for deployment on ARM, Intel, and Nvidia processors. (*Before Neo, you will need to spend major man-hour efforts to deploy AI/ML Models on a specific hardware with specific compiler, memory, operating systems…etc)*.

DEEP LEARNING ON AWS

DEEP LEARNING ON AWS EC2

- AWS Deep Learning AMIs provide AI/ML developers with the infrastructure to quickly develop and scale deep learning in the cloud.
- Amazon EC2 instance could be easily launched on Amazon Linux or Ubuntu. The instance comes readily available with all the deep learning frameworks.
- Frameworks such as Apache MXNet, TensorFlow, the Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Torch and Keras.
- These tools can be used to build and train advanced AI/ML models.
- Instance types for deep learning include:
    - P3: 8 Tesla V100 GPU's
    - P2: 16 K80 GPU's
    - G3: 4 M60 GPU's (all Nvidia chips)

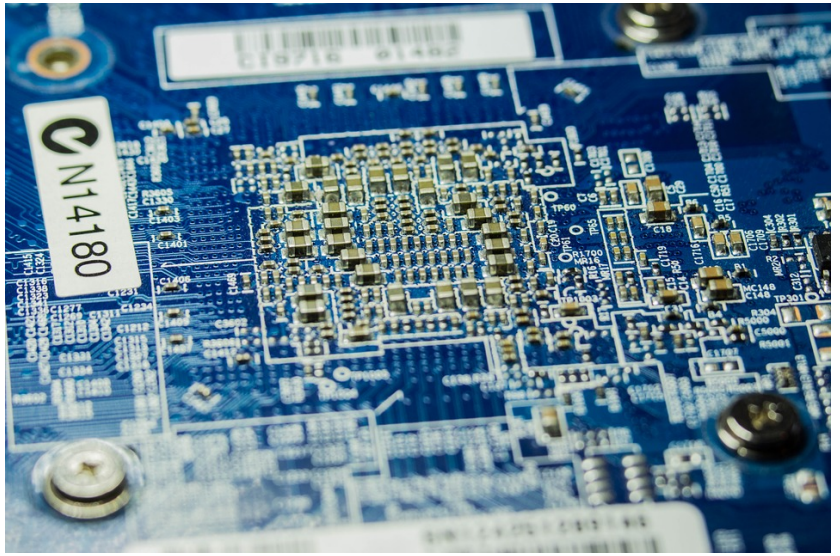


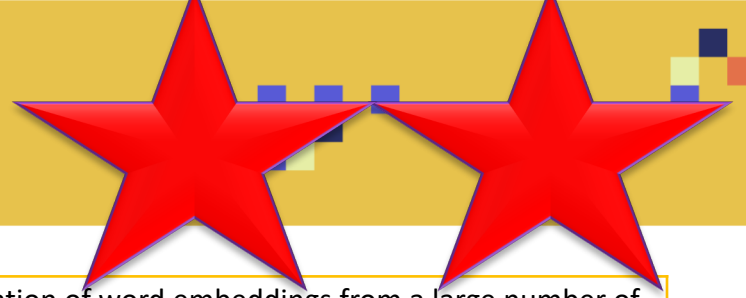
Photo Credit: https://pixabay.com/photos/transistors-gpu-processor-pc-chip-1137502/

DEEP LEARNING ON AWS EMR (ELASTIC MAP REDUCE)

- EMR supports Apache MXNet and GPU instance types
- Recall that:
  - Amazon Elastic MapReduce (EMR) is an AWS tool for big data processing.
  - Amazon EMR allows developers to build an expandable low configuration service at a fraction of the cost of using an in-house cluster computing.
  - Amazon EMR is capable of processing big data across a Hadoop cluster of virtual servers on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3).
  - The term elastic means that EMR offers dynamic resizing ability meaning it could increase or decrease resources based on the demand.

AWS SAGEMAKER BUILT-IN ALGORITHMS

SAGEMAKER AVAILABLE ALGORITHMS



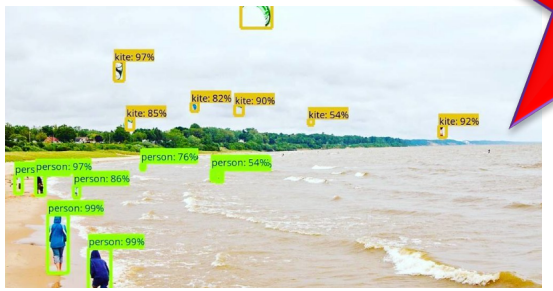
BlazingText Word2Vec	BlazingText implementation of the Word2Vec algorithm for scaling and accelerating the generation of word embeddings from a large number of documents.
DeepAR	An algorithm that generates accurate forecasts by learning patterns from many related time-series using recurrent neural networks (RNN).
Factorization Machines	A model with the ability to the estimate all of the interactions between features even with a very small amount of data.
Gradient Boosted Trees (XGBoost)	Short for “Extreme Gradient Boosting”, XGBoost is an optimized distributed gradient boosting library.
Image Classification (ResNet)	A popular neural network for developing image classification systems.
IP Insights	An algorithm to detect malicious users or learn to usage patterns of IP addresses.
K-Means Clustering	One of the simplest ML algorithms. It’s used to find groups within unlabeled data.
K-Nearest Neighbor (k-NN)	An index based algorithm to address classification and regression based problems.
Latent Dirichlet Allocation (LDA)	A model that is well suited to automatically discovering the main topics present in a set of text files.
Linear Learner (Classification)	Linear classification uses an object’s characteristics to identify the appropriate group that it belongs to.
Linear Learner (Regression)	Linear regression is used to predict the linear relationship between two variables.
Neural Topic Modelling (NTM)	A neural network based approach for learning topics from text and image datasets.
Object2Vec	A neural-embedding algorithm to compute nearest neighbors and to visualize natural clusters.
Object Detection	Detects, classifies, and places bounding boxes around multiple objects in an image.
Principal Component Analysis (PCA)	Often used in data pre-processing, this algorithm takes a table or matrix of many features and reduces it to a smaller number of representative features.
Random Cut Forest	An unsupervised machine learning algorithm for anomaly detection.
Semantic Segmentation	Partitions an image to identify places of interest by assigning a label to the individual pixels of the image.
Seqence2Sequence	A general-purpose encoder-decoder for text that is often used for machine translation, text summarization, etc.

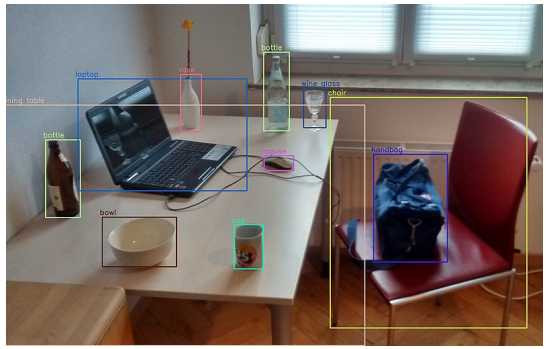
Source: https://aws.amazon.com/sagemaker/build/

OBJECT DETECTION

OBJECT DETECTION: OVERVIEW

- SagMaker object detection uses deep learning to:
  - Detects objects
  - Classify them
- Object detection is a supervised training algorithm that is capable of detecting multiple objects in a given image and classifying them.
- The algorithm generates a bounding box along with a confidence interval associated with each object.
- The algorithm uses Single Shot multibox Detector (SSD) framework and supports two base networks:
  - VGG (Visual Geometry Group)
  - ResNet (Residual Network)
- You can start training the network from scratch of start from a pretrained network using transfer learning.





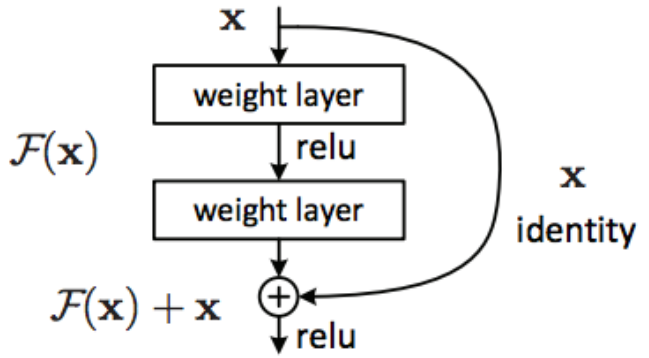
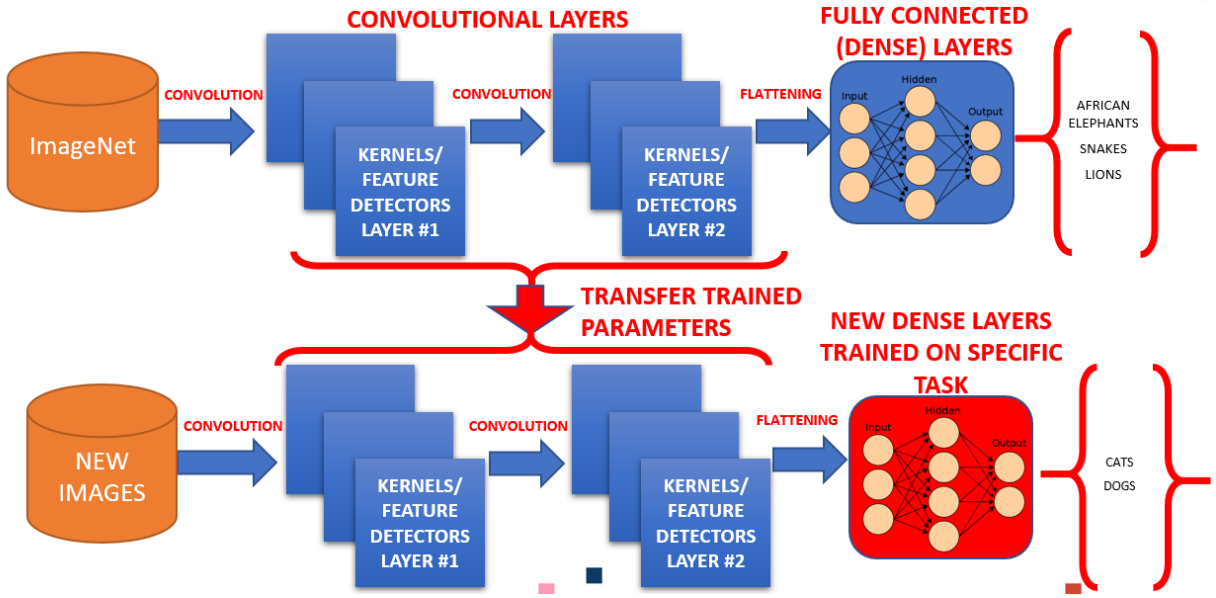


Photo Credit: https://commons.wikimedia.org/wiki/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg
Photo Credit: https://www.flickr.com/photos/drbeachvacation/35477618781
Photo Credit: https://commons.wikimedia.org/wiki/File:Resnet.png

OBJECT DETECTION: INCREMENTAL TRAINING

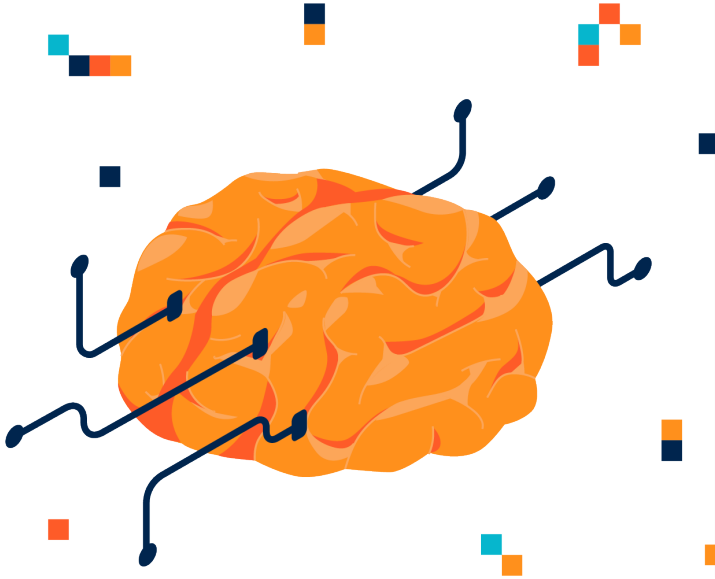
- Incremental training allows developers to start from a pretrained model instead of training a model from scratch.
- Incremental training is extremely efficient and saves time and resources.
- You can perform image augmentation to avoid model overfitting such as flip, rescale, and jitter



OBJECT DETECTION: INPUT/OUTPUT

- Object detection algorithm supports the following:
  - RecordIO (application/x-recordio) (for file mode)
  - Image (image/png, image/jpeg, and application/x-image) (for file mode).
- Note: for training in pipe mode, RecordIO (application/x-recordio) should be used.
- Each image needs a .json file for annotation, and the .json file should have the same name as the corresponding image.
- 4 properties as follows:
  - "file" specifies the path of the image file.
  - "image_size" property indicates image dimensions.
  - "annotations" property indicates the categories and bounding boxes for objects within the image.
  - "categories" includes mapping between class index and class name.

```
{ "file": "your_image_directory/sample_image1.jpg", "image_size":
[
{ "width": 500,
  "height": 400,
  "depth": 3
}
],

"annotations": [

{ "class_id": 0,
  "left": 111,
  "top": 134,
  "width": 61,
  "height": 128
},

{ "class_id": 0,
"left": 161,
"top": 250,
"width": 79,
"height": 143
},

{ "class_id": 1,
  "left": 101,
  "top": 185,
  "width": 42,
  "height": 130 } ],

"categories": [
{ "class_id": 0,
  "name": "dog" },
{ "class_id": 1,
"name": "cat" } ] }
```



OBJECT DETECTION: HYPERPARAMETERS

- **base_network:** The base network architecture to use.
- use_pretrained_model: Indicates whether to use a pre-trained model for training.
- num_classes: The number of output classes.
- image_shape: The image size for input images.
- freeze_layer_pattern: The regular expression (regex) for freezing layers in the base network.
- Mini_batch_size
- Learning_rate
- Optimizer: Sgd, adam, rmsprop, adadelta

For full list of hyperparameters: https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection-api-config.html

OBJECT DETECTION: EC2 INSTANCE

- For training:
  - GPU instances for training: ml.p2.xlarge, ml.p2.8xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.8xlarge and ml.p3.16xlarge.
  - When training with large batch sizes, GPU instances with more memory is recommended.
  - You can also run the algorithm on multi-GPU and multi-machine settings for distributed training.
- For inference:
  - CPU (C5 and M5) and GPU (P2 and P3) instances can be used.

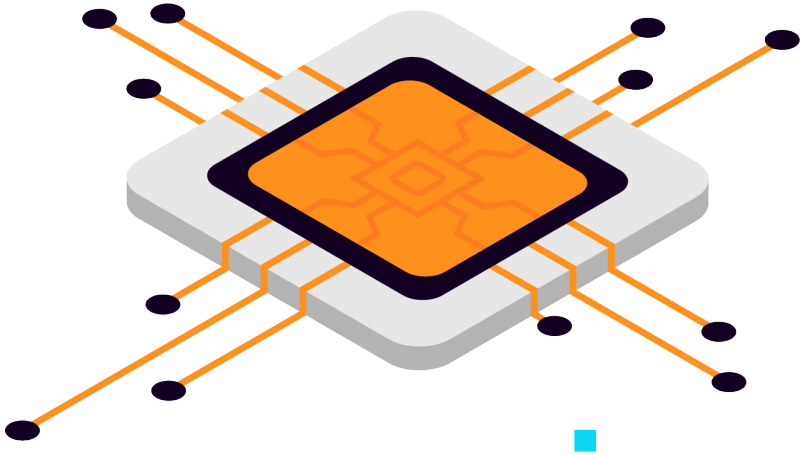


IMAGE CLASSIFICATION

IMAGE CLASSIFICATION: OVERVIEW

- Image classification algorithm is a supervised machine learning algorithm
- The algorithm is capable for classifying multiclass images.
- The algorithm DOES NOT specify the location of the object in the image so there is no bounding box.
- The algorithm takes in an image as an input
- It generates the label as an output such as: *Stop Sign, deer crossing, yield,..etc*
- SageMaker image classification algorithm uses convolutional neural network known as ResNet to perform classification.
- The network could be trained from scratch or starting from a pretrained network using transfer learning.

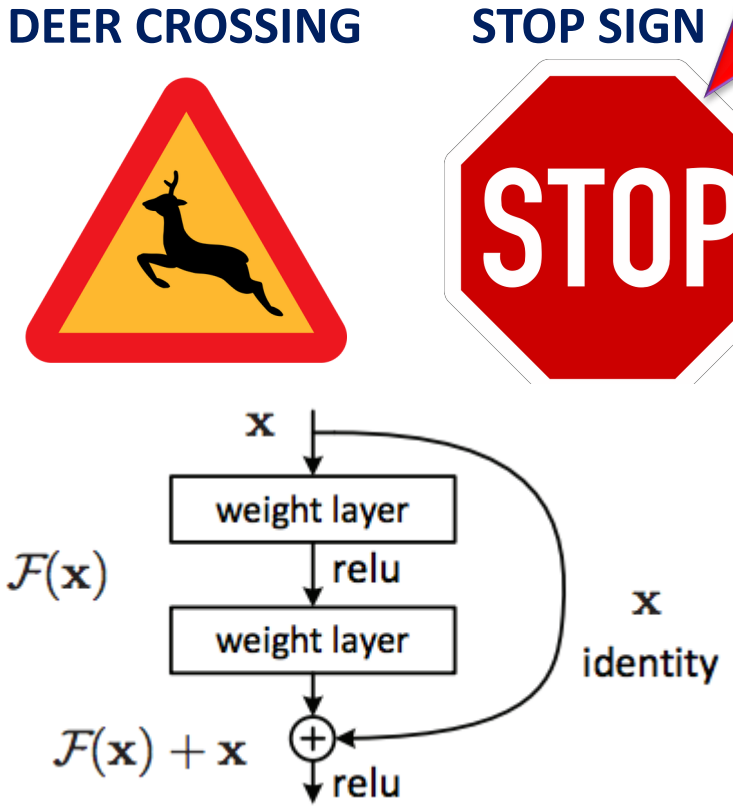
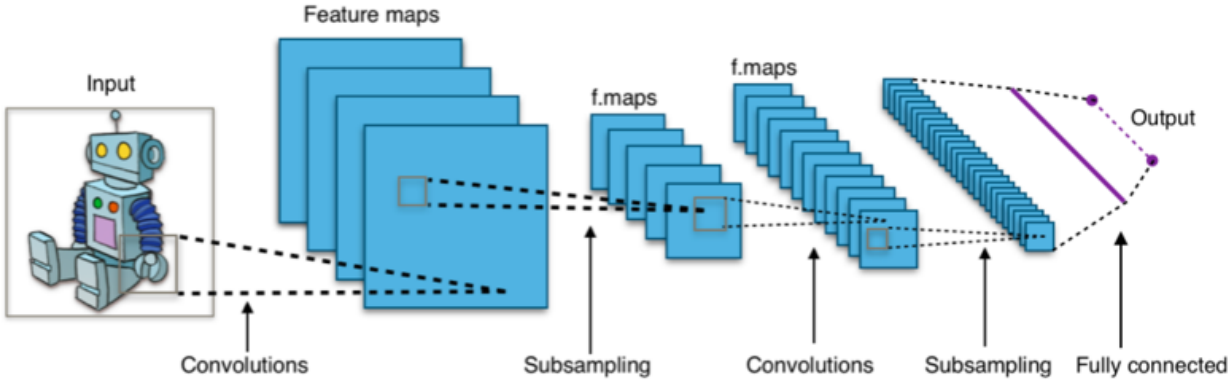
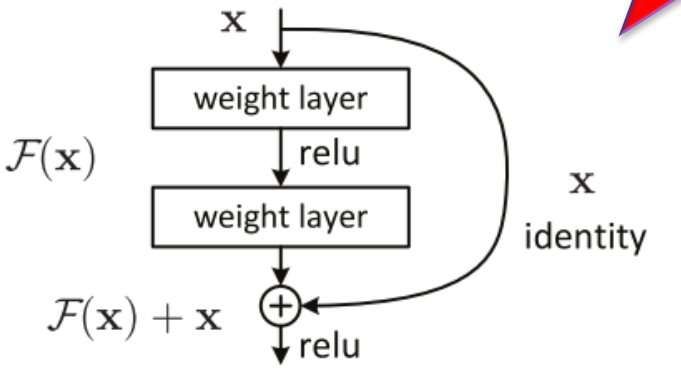


Photo Credit: https://pixabay.com/illustrations/traffic-sign-road-sign-shield-6627/
Photo Credit: https://publicdomainvectors.org/en/free-clipart/Warning-for-deer-traffic-sign-vector/4025.html
Photo Credit: https://commons.wikimedia.org/wiki/File:Resnet.png

IMAGE CLASSIFICATION: DEEPDIVE





- Great article: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
- ImageNet is a large dataset with over 11 million images consisting of 11,000 categories.
- ImageNet is used to train ResNet deep network
- Once the networks is trained with ImageNet, the network can be repurposed for specific task using transfer learning.
- There are two modes for Image classification in Amazon SageMaker: (1) full training, (2) transfer learning.
  - In full training mode: network is being trained from scratch and with randomly initialised weights.
  - In transfer learning model: network is initialized with pre-trained weights and the classification head (dense layer) is initialized with random weights. The network is being trained with the new dataset which could be small.
- Expected Image size is 224x224x3

Photo Credit: https://en.wikipedia.org/wiki/File:Typical_cnn.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Resnet.png

IMAGE CLASSIFICATION: INPUT/OUTPUT

- Amazon SageMaker recommends Apache MXNet RecordIO for the image input.
- The algorithm supports both RecordIO (application/x-recordio) and image (image/png, image/jpeg, and application/x-image) content types for training in file mode
- The algorithm supports RecordIO (application/x-recordio) content type for training in pipe mode.
- Augmented Manifest Image Format enables Pipe mode
- The algorithm supports image/png, image/jpeg, and application/x-image for inference.
- Image format requires .lst files to associate image index, class label, and path to the image.

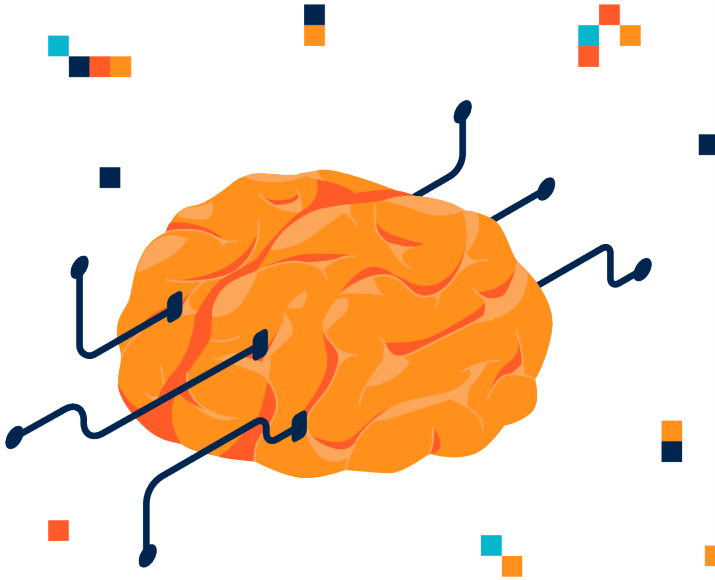


IMAGE INDEX **CLASS LABEL** **PATH TO IMAGE FILE**

5 1 your_image_directory/train_img_dog1.jpg 1000 0
your_image_directory/train_img_cat1.jpg 22 1
your_image_directory/train_img_dog2.jpg

IMAGE CLASSIFICATION: HYPERPARAMETERS

- For the full list of hyperparameters: https://docs.aws.amazon.com/sagemaker/latest/dg/IC-Hyperparameter.html
- use_pretrained_model: Indicates whether to use a pre-trained model for training.
- num_classes: The number of output classes.
- augmentation_type: Data augmentation type
- image_shape: The image size for input images.
- Mini_batch_size
- Weight decay
- beta 1
- beta 2
- eps
- gamma
- Learning_rate
- Optimizer: Sgd, adam, rmsprop, adadelta

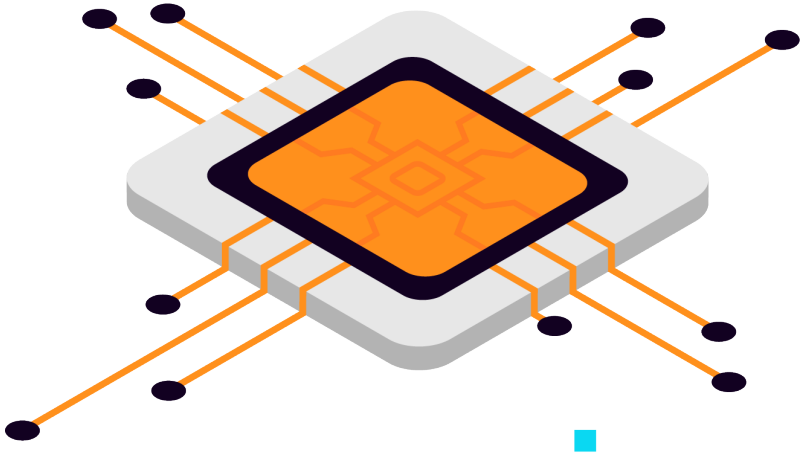
IMAGE CLASSIFICATION: EC2 INSTANCE

For training:

- GPU instances for training: ml.p2.xlarge, ml.p2.8xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.8xlarge and ml.p3.16xlarge.
- When training with large batch sizes, GPU instances with more memory is recommended.
- You can also run the algorithm on multi-GPU and multi-machine settings for distributed training.

For inference:

- CPU (C5 and M5) and GPU (P2 and P3) instances can be used.



SEMANTIC SEGMENTATION

SEMANTIC SEGMENTATION: OVERVIEW

- Semantic segmentation algorithm provides image classification on the pixel-level.
- Given a set of classes, SageMaker semantic segmentation algorithm tags every pixel with a class label.
- Semantic segmentation is critical for computer vision applications such as self-driving cars.
- Recall that:
  - **(1) Amazon SageMaker Image Classification:** was a supervised learning algorithm that takes in an entire image and classify it one or more classes.
  - **(2) Object Detection Algorithm:** was a supervised algorithm that **detects and classifies** all objects in a given image by providing a bounding box around the object.

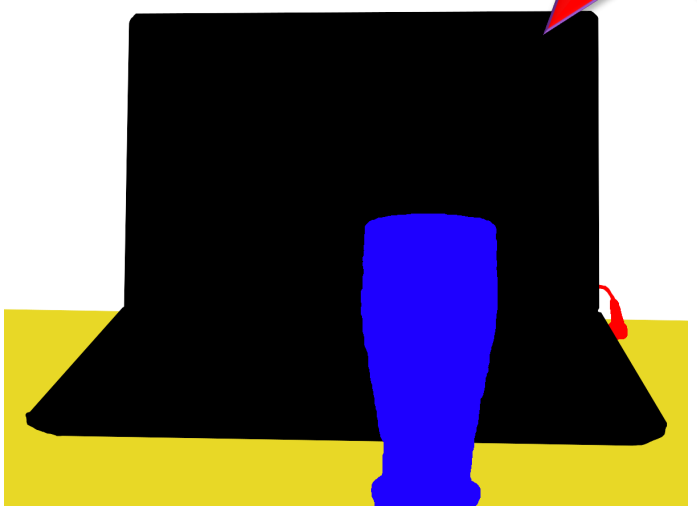


Photo Credit: https://commons.wikimedia.org/wiki/File:Image-segmentation-example-segmented.png

SEMANTIC SEGMENTATION: OVERVIEW

- One of the key advantages of semantic segmentation is that it provides details about the shape of the object because it looks at every pixel.
- The segmentation algorithm provides a segmentation mask which is a RGB (or grayscale) image with the same shape as the input image.
- Built using MXNet Gluon framework and Gluon CV toolkit.
- Three deep neural network-based algorithms are available:
  - Fully-Convolutional Network (FCN) algorithm
  - Pyramid Scene Parsing (PSP) algorithm
  - DeepLabV3
- The algorithm consists of two elements:
  - *Encoder (backbone)*: network that generates activation maps of features.
  - *Decoder:* network that takes in the encoded activation maps and constructs segmentation mask from it.

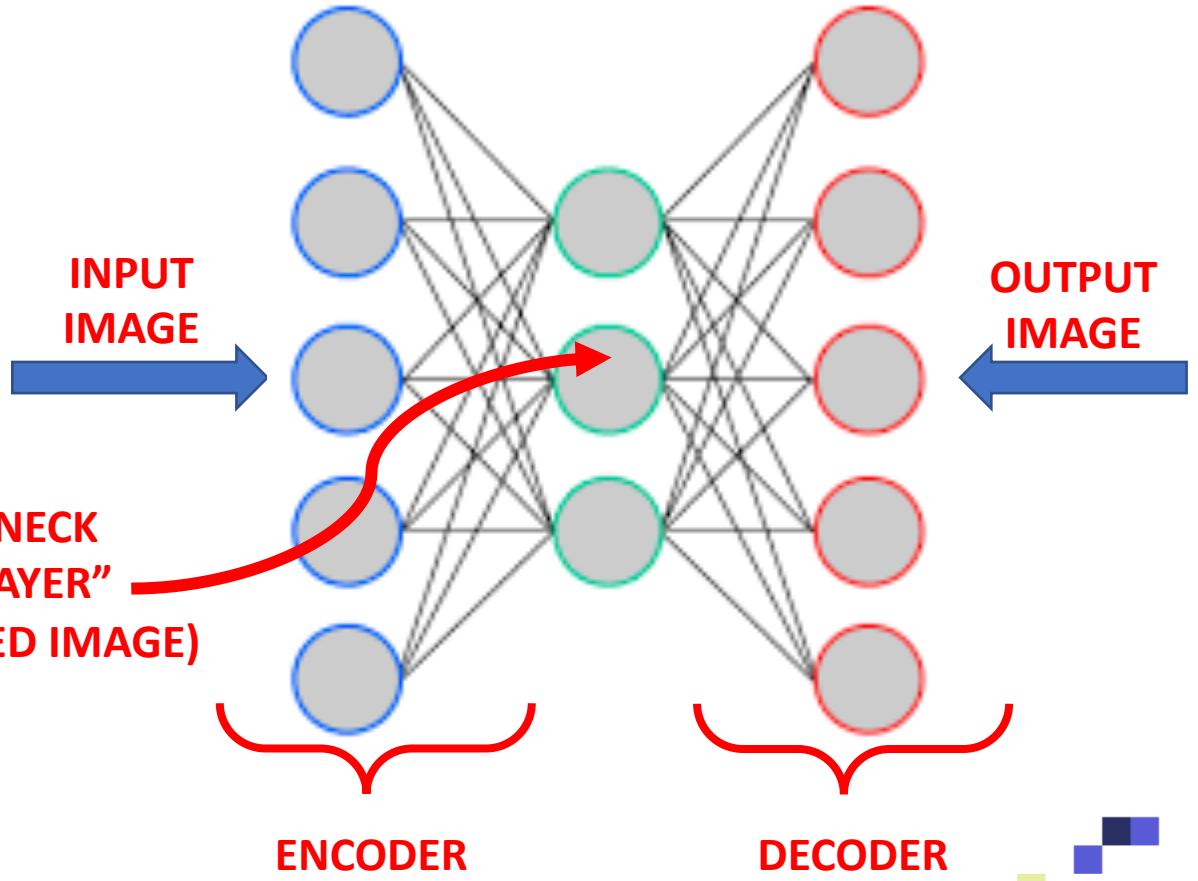
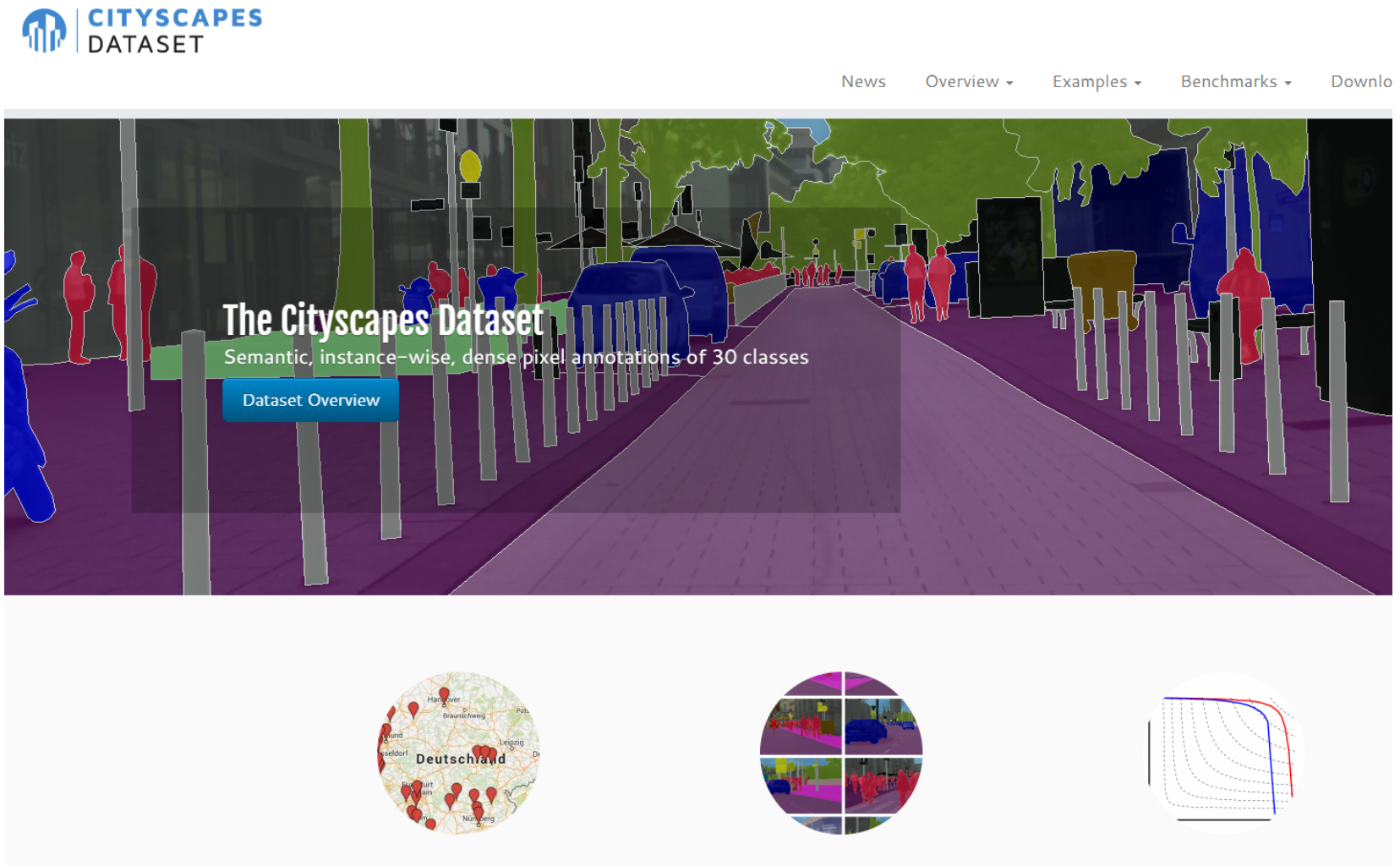


Photo Credit: https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network_image_recognition.png

SEMANTIC SEGMENTATION: OVERVIEW

- Several backbones are available for the FCN, PSP, and DeepLabV3 algorithms:
  - ResNet50
  - ResNet101
- Backbone networks are trained using ImageNet
- Backbones can be trained from scratch or using pretrained network (transfer learning)
- Decoders MUST be trained from scratch.
- For inference:
  - Amazon SageMaker hosting service is used to deploy trained model.
  - Segmentation mask can be a PNG image or set of probabilities for each class for each pixel.

SEMANTIC SEGMENTATION: CITYSCAPES DATASET



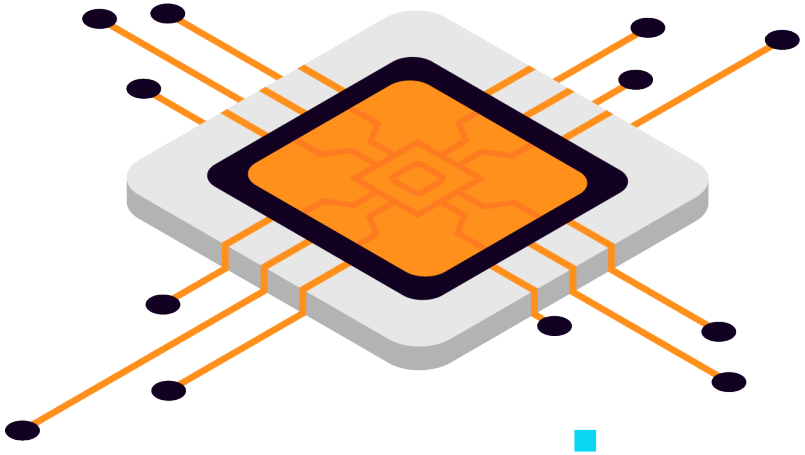
https://www.cityscapes-dataset.com/

SEMANTIC SEGMENTATION: HYPERPARAMETERS

- For the entire list of hyperparameters, check this out: https://docs.aws.amazon.com/sagemaker/latest/dg/segmentation-hyperparameters.html
- Backbone: The backbone to use for the algorithm's encoder component, examples: resnet-50, resnet-101, resnet-50
- use_pretrained_model: Whether a pretrained model is to be used for the backbone.
- Algorithm: The algorithm to use for semantic segmentation, examples: fcn: Fully-Convolutional Network (FCN) algorithm, psp: Pyramid Scene Parsing (PSP) algorithm, deeplab: DeepLab V3 algorithm
- num_classes: number of classes to segment.
- learning rate
- batch size
- Epochs
- optimizer

SEMANTIC SEGMENTATION: EC2 INSTANCE

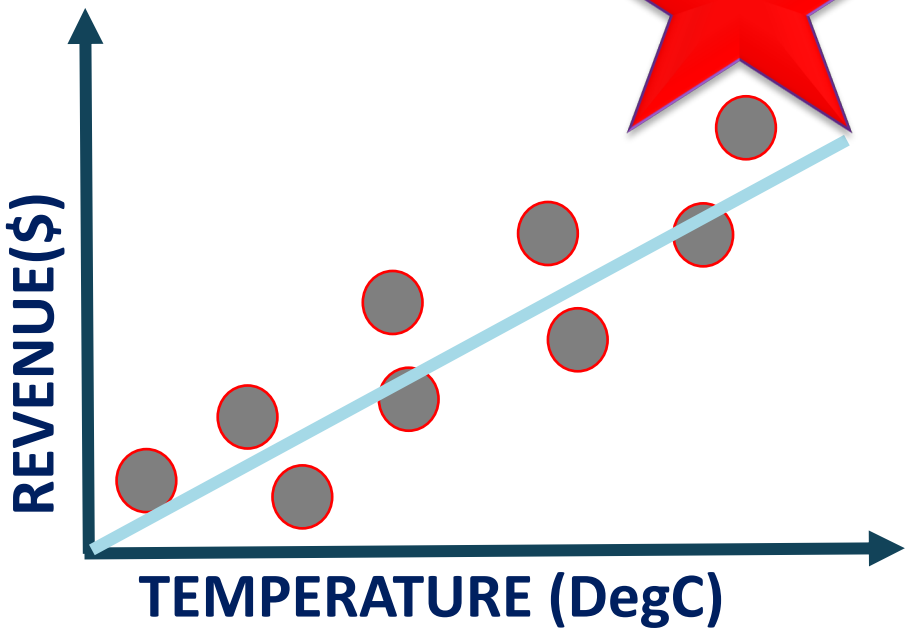
- For training:
  - GPU P2 or P3
  - GPU instances for training: ml.p2.xlarge, ml.p2.8xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.8xlarge and ml.p3.16xlarge.
- For inference:
  - CPU (C5 and M5) and GPU (P2 and P3) instances can be used.



SAGEMAKER LINEAR LEARNER

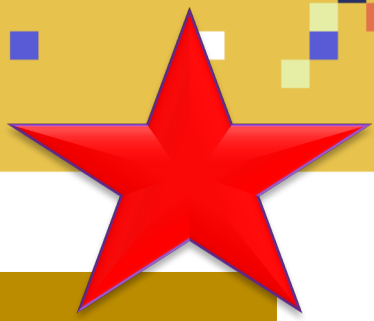
SAGEMAKER LINEAR LEARNER: OVERVIEW

- Linear Learner is a supervised learning algorithm that is used to fit a line to the training data.
- It could be used for both classification and regression tasks as follows:
  - **Regression:** output contains continuous numeric values
  - **Binary classification:** output label must be either 0 or 1 (linear threshold function is used).
  - **Multiclass classification:** output labels must be from 0 to *num_classes - 1*.
- The best model optimizes either of the following:
  - **For regression:** focus on Continuous metrics such as mean square error, root mean squared error, cross entropy loss, absolute error.
  - **For classification:** focus on discrete metrics such as F1 score, precision, recall, or accuracy.



	Temperature	Revenue
0	24.566884	534.799028
1	26.005191	625.190122
2	27.790554	660.632289
3	20.595335	487.706960
4	11.503498	316.240194
5	14.352514	367.940744
6	13.707780	308.894518
7	30.833985	696.716640
8	0.976870	55.390338
9	31.669465	737.800824
10	11.455253	325.968408
11	3.664670	71.160153

SAGEMAKER LINEAR LEARNER: USE CASES



DISCRETE BINARY CLASSIFICATION

- Does this patient have a disease or not?

DISCRETE MULTICLASS CLASSIFICATION

- Should an autonomous car stop, slow down or accelerate?

REGRESSION TASKS

- Revenue predictions based on previous years performance.

SAGEMAKER LINEAR LEARNER: OVERVIEW

Preprocessing

- Ensure that data is shuffled before training
- Normalization or feature scaling is offered by Linear Learner (which is great!)
- Normalization or feature scaling is a critical preprocessing step to ensure that the model does not become dominated by the weight of a single feature.

Training

- Linear Learner uses stochastic gradient descent to perform the training
- Select an appropriate optimization algorithm such as Adam, AdaGrad, and SGD
- Hyperparameters, such as momentum, learning rate, and the learning rate schedule can be selected.
- Overcome model overfitting using L1, L2 regularization
- Multiple models could be optimized in parallel

Validation

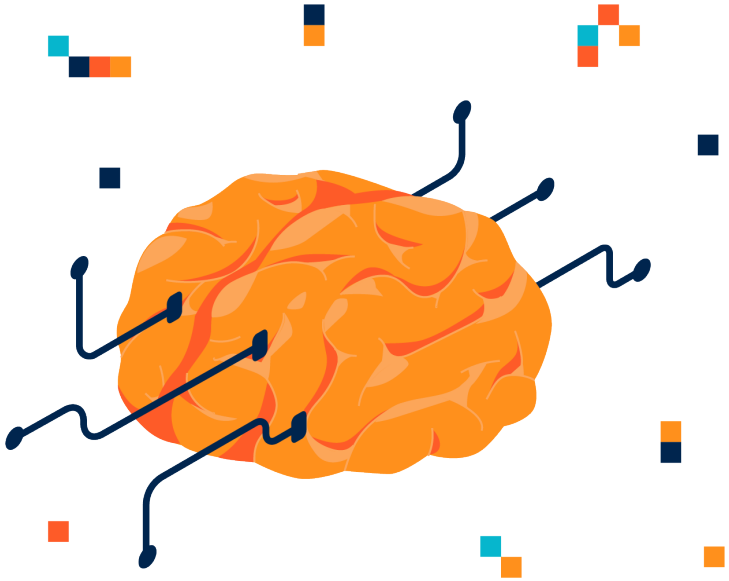
- Trained models are evaluated against a validation dataset and best model is selected based on the following metrics:
    - **For regression:** mean square error, root mean squared error, cross entropy loss, absolute error.
    - **For classification:** F1 score, precision, recall, or accuracy.

SAGEMAKER LINEAR LEARNER HYPERPARAMETERS

- **Learning Rate:** The step size used by the optimizer for parameter updates.
- **L1:** L1 regularization parameter.
- **Momentum:** momentum of the SGD optimizer.
- **Mini_batch_size:** The number of observations per mini-batch
- **Wd:** The weight decay parameter, also known as the L2 regularization parameter.
- Check out the rest of hyperparameters here: https://docs.aws.amazon.com/sagemaker/latest/dg/ll_hyperparameters.html

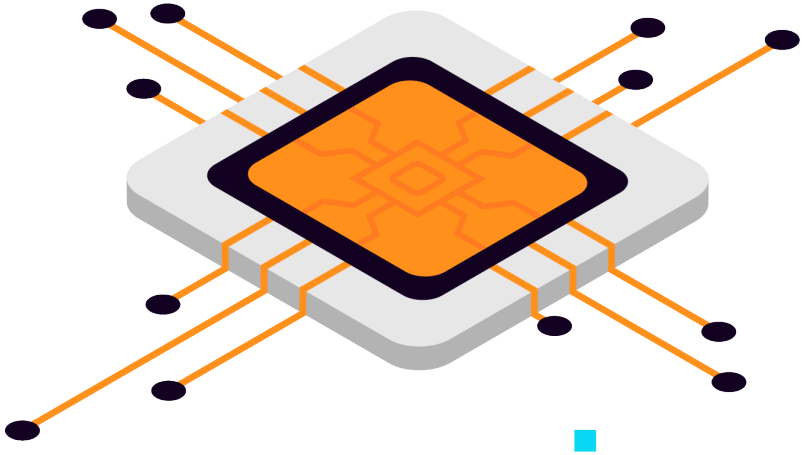
SAGEMAKER LINEAR LEARNER: INPUT/OUTPUT DATA

- Amazon SageMaker linear learner supports the following input data types:
  - RecordIO-wrapped protobuf (*note: only Float32 tensors are supported*)
  - Text/CSV (*note: First column assumed to be the target label*)
  - File or Pipe mode both supported
- For inference, linear learner algorithm supports the application/json, application/x-recordio-protobuf, and text/csv formats.
- For regression (predictor_type='regressor'), the score is the prediction produced by the model.
- For classification (predictor_type='binary_classifier' or predictor_type='multiclass_classifier'), the model returns a score and also a predicted_label. The predicted_label is the class predicted by the model and the score measures the strength of that prediction.



SAGEMAKER LINEAR LEARNER: EC2 INSTANCE

- Linear Learner algorithm could be trained on:
  - **Single CPU and GPU instances**
  - **Multi-Machine CPU and GPU instances**
- During testing, multi-GPU computers are not necessary (add cost with no value).



FACTORIZATION MACHINES

FACTORIZATION MACHINES: OVERVIEW

- A factorization machine is a supervised learning algorithm.
- It could be used to perform general purpose classification and regression operations.
- It is an extension of a linear model and works well with highly sparse data.
- An example of high sparse data is (1) click prediction and (2) item recommendation.
- For example, factorization machines can be used to predict the behaviour of customers with an ad is placed by tracking the number/rate of clicks patterns. This is a prime example of sparse dataset.

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	I
user_id																		
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	2.0	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
6	NaN	NaN	NaN	4.0	NaN	NaN	NaN	5.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
7	NaN	NaN	NaN	4.0	NaN	NaN	5.0	5.0	NaN	4.0	...	NaN	NaN	NaN	5.0	3.0	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	NaN	NaN	NaN	NaN	
10	NaN	NaN	NaN	5.0	NaN	NaN	NaN	5.0	NaN	4.0	...	NaN	NaN	NaN	NaN	NaN	NaN	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
13	NaN	NaN	2.0	4.0	NaN	NaN	2.0	5.0	1.0	4.0	...	NaN	2.0	NaN	5.0	3.0	NaN	
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	

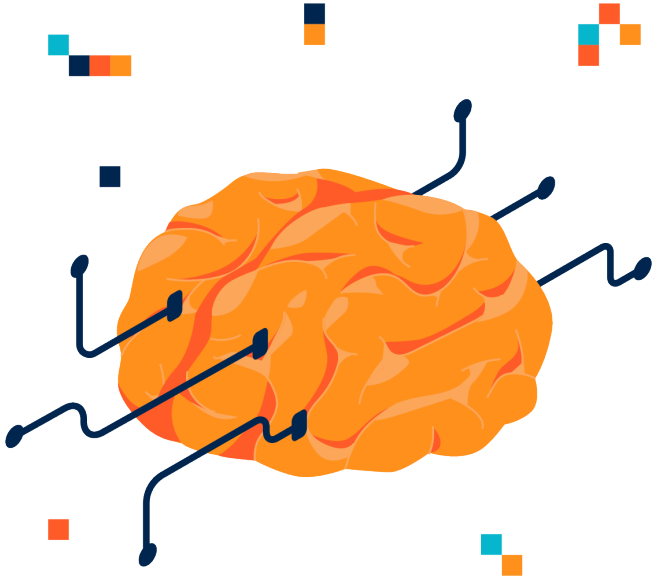
SPARSE DATASET WHERE USERS DON'T GENERALLY RATE EVERY MOVIE OR PRODUCT!

FACTORIZATION MACHINES: HYPERPARAMETERS

- Full set of hyperparameters: https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines-hyperparameters.html
- feature_dim: number of features in the input data.
- num_factors: dimensionality of factorization.
- predictor_type: type of predictor, either binary_classifier or regressor.
- bias_init_method: initialization method for bias term.
  - Normal: weights are initialized according to normal distribution.
  - uniform: weights initialized with uniform distribution.
  - constant: weights initialized to scalar values.

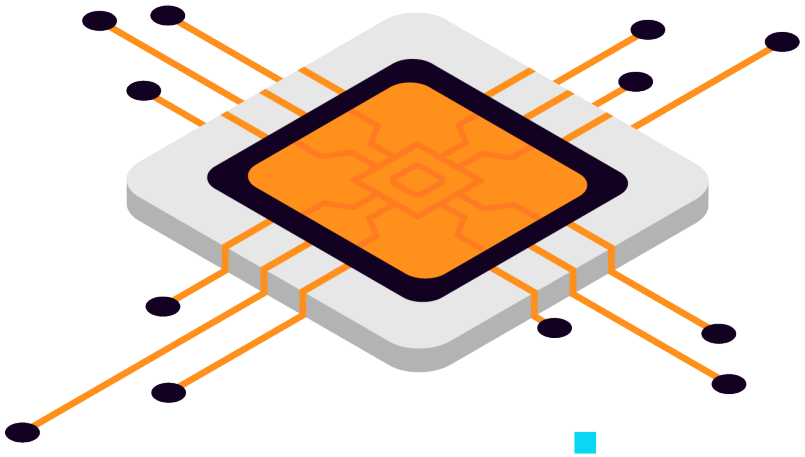
FACTORIZATION MACHINES: INPUT/OUTPUT

- During training:
  - factorization machines expects recordIO-protobuf data format with Float32 tensors.
  - CSV does not work well because data is sparse.
  - Both File and Pipe mode training are supported for recordIO-wrapped protobuf.
- During inference:
  - Factorization machines support the application/json and x-recordio-protobuf formats.



FACTORIZATION MACHINES: INSTANCE TYPES

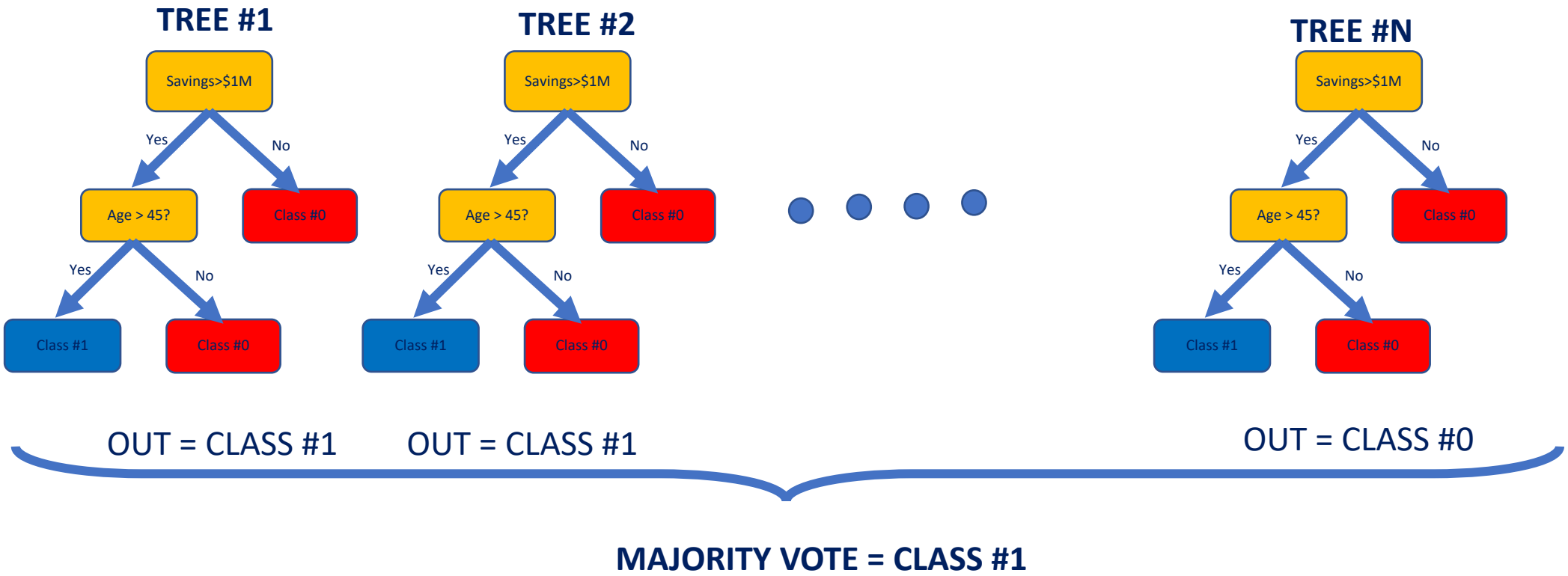
- SageMaker Factorization Machines is highly scalable.
- It could work on multiple distributed instances.
- For Factorization Machines Training and inference:
  - CPU instance is recommended since GPU is preferred with dense data only



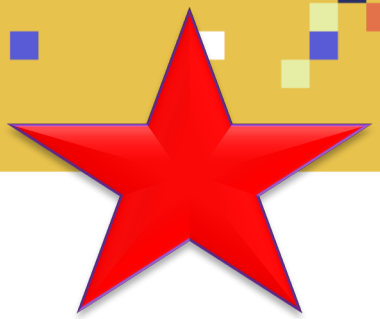
SAGEMAKER XGBOOST

SAGEMAKER XGBOOST: OVERVIEW

- XGBoost or Extreme Gradient Boosting algorithm is one of the most famous and powerful algorithms to perform both regression and classification tasks.
- XGBoost is a supervised learning algorithm and implements gradient boosted trees algorithm.
- The algorithm work by combining an ensemble of predictions from several weak models.



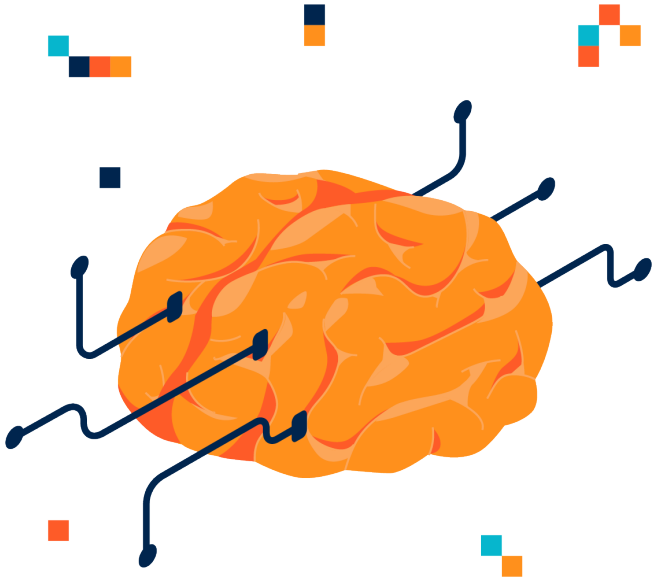
SAGEMAKER XGBOOST: OVERVIEW



- Recently, XGBoost is the go to algorithm for most developers and has won several Kaggle competitions.
- Why does Xgboost work really well?
  - Since the technique is an ensemble algorithm, it is very robust and could work well with several data types and complex distributions.
  - Xgboost has a many tunable hyperparameters that could improve model fitting.
- What are the applications of XGBoost?
  - XGBoost could be used for fraud detection to detect the probability of a fraudulent transactions based on transaction features.

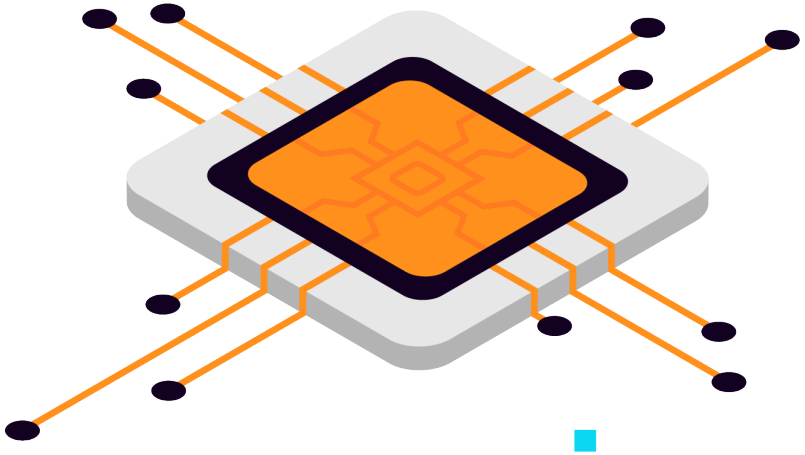
SAGEMAKER XGBOOST: INPUT/OUTPUT DATA

- Gradient boosting uses tabular data for inputs/outputs:
  - Rows represent observations,
  - One column represents the output or target label
  - The rest of the columns represent the inputs (features)
- Amazon SageMaker implementation of XGBoost supports the following file format for training and inference :
  - CSV
  - libsvm
- Xgboost does not support protobuf format (*note: this is unique compared to other Amazon SageMaker algorithms, which use the protobuf training input format*).



SAGEMAKER XGBOOST: EC2 INSTANCE

- XGBoost currently only trains using CPUs.
- XGboost is **memory intensive algorithm** so it does not require much compute.
- M4: General-purpose compute instance is recommended.



SAGEMAKER XGBOOST: HYPERPARAMETERS

- **Alpha:** L1 regularization term on weights.
- **Lambda:** L2 regularization
- **Booster:** Which booster to use.
- **Eta:** Step size shrinkage used in updates to prevent overfitting.
- **Gamma:** Minimum loss reduction needed to add more partitions to the tree.
- Check out the rest of hyperparameters here: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost_hyperparameters.html

SAGEMAKER SEQ2SEQ

SAGEMAKER SEQUENCE-TO-SEQUENCE: OVERVIEW

- Sequence to Sequence is a supervised machine learning algorithm available in SageMaker.
- Under the hood, the algorithm uses Recurrent Neural Networks (RNNs) and Convolutional Neural Network (CNN).
- The algorithm takes in sequence of tokens such as audio and generate a sequence of tokens as well.
- Examples:
  - **Machine Translation:**
    - Input: English sentence
    - output: translated French sentence
  - **Text summarization:**
    - Input: Long sentence
    - Output: summarized sentence
  - **Speech to text:**
    - Input: audio clips
    - Output: tokens of words

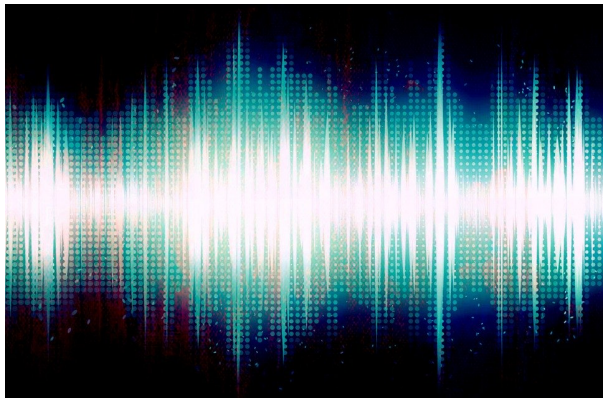


Photo Credit: https://pixabay.com/illustrations/speech-icon-voice-talking-audio-2797263/
Photo Credit: https://pixabay.com/illustrations/sound-audio-waves-equalizer-495859/

SAGEMAKER SEQUENCE-TO-SEQUENCE: HOW DOES IT WORK?

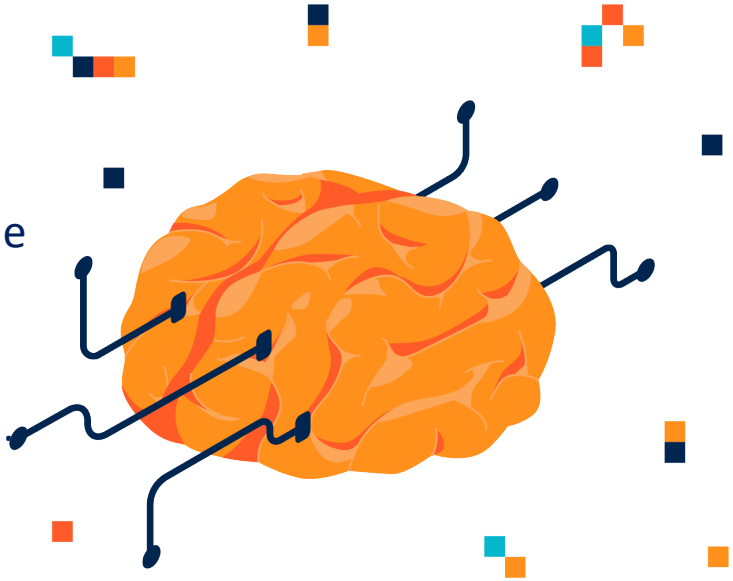
- Seq2Seq consists of the following layers:
  - **Embedding layer:** the sparse one hot encoded input is being mapped to dense feature layer.
  - **Encoder layer:**
    - Consists of LSTM or GRU
    - It takes input sequence from pervious layer and compress the information into a fixed-length feature vector.
  - **Decoder layer:**
    - Consists of RNN and LSTM
    - Decoder layer takes this encoded feature vector and generates output sequence of tokens.

SAGEMAKER SEQUENCE-TO-SEQUENCE: HOW DOES IT WORK?

- Seq2Seq is based on Attention mechanism.
- Attention mechanism has been developed to overcome the issues exist with the traditional encoder-decoder framework.
- As the length of the dataset increases, the performance of the network becomes poor.
- This is because of the limit of how much information the fixed-length encoded feature vector can contain.
- Attention mechanism overcomes these problems by enabling the decoder to locate the most important information in the encoder so it could predict the next token in the sequence.

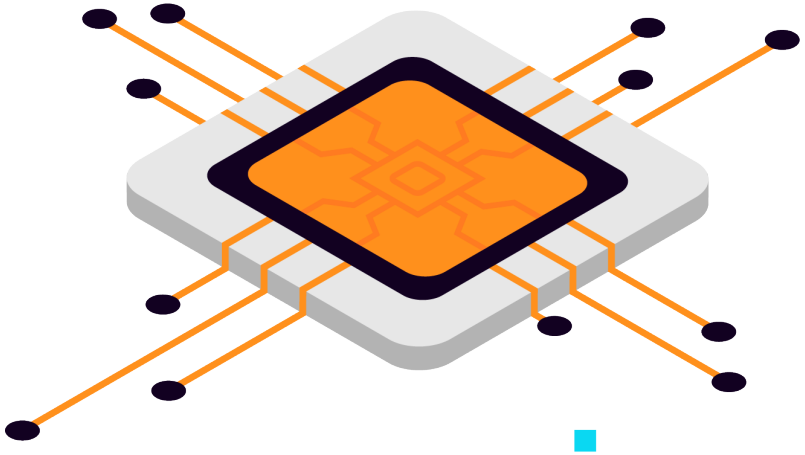
SAGEMAKER SEQUENCE-TO-SEQUENCE: INPUT/OUTPUT

- Seq2seq accepts data in RecordIO-Protobuf format.
- Tokens must be integers format which is an exception from the floating point formatting norm.
- You can use a readily available script by SageMaker that converts from tokenized text files to protobuf.
- After you complete preprocessing stage, the algorithm can be called to kick off the training process.
- The algorithm expects 3 channels as follows:
  - **train:** contains training data (train.rec file generated by preprocessing script).
  - **validation:** contains validation data (val.rec file generated by preprocessing script).
  - **vocab:** contains two vocabulary files (vocab.src.json and vocab.trg.json)



SAGEMAKER SEQUENCE-TO-SEQUENCE: EC2 INSTANCES

- Currently Amazon SageMaker seq2seq is only supported on GPU instance types
- SageMaker is only set up to train on a single machine.
- But it does also offer support for multiple GPUs.



SAGEMAKER SEQUENCE-TO-SEQUENCE: HYPERPARAMETERS

- Check out the full list of parameters here: https://docs.aws.amazon.com/sagemaker/latest/dg/seq-2-seq-hyperparameters.html
- Batch_size: Mini batch size for gradient descent
- cnn_activation_type: CNN activation function type.
- encoder_type: Encoder type. RNN architecture/CNN Architectures
- Optimizer_type (adam, sgd, rmsprop)
- Learning_rate: Initial learning rate.
- Num_layers_encoder: Number of layers for Encoder rnn or cnn.
- Num_layers_decoder: Number of layers for Decoder *rnn* or *cnn*.

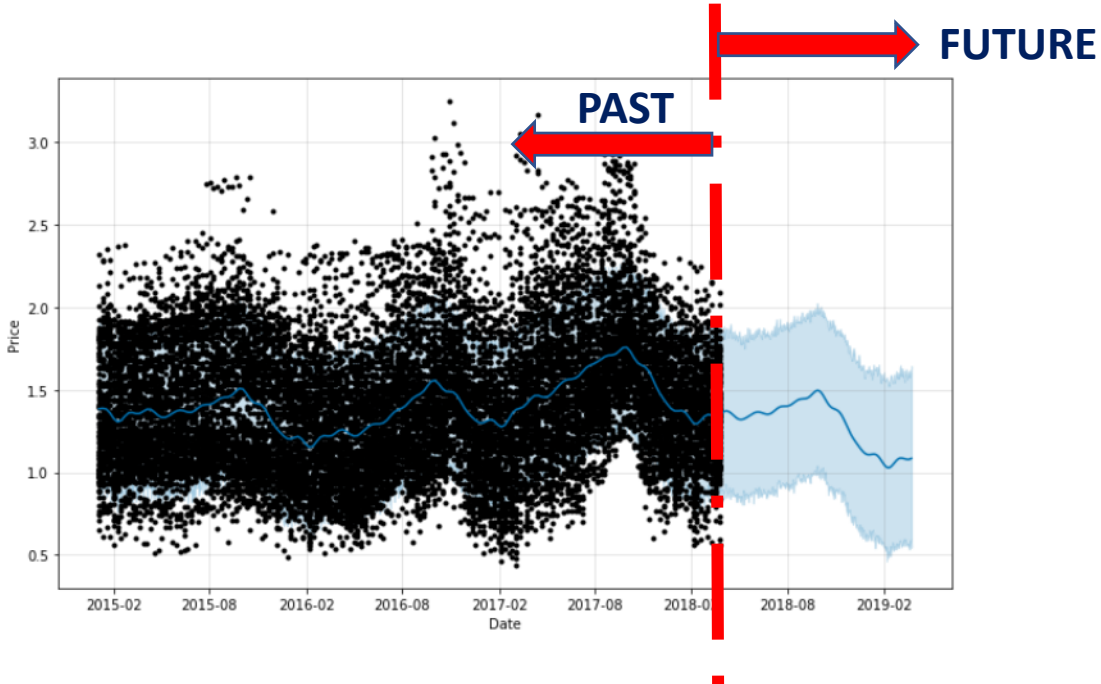
DEEPAR

DEEPAR: OVERVIEW

- Amazon SageMaker DeepAR is a one dimensional time series forecasting algorithm
- It works in supervised fashion and uses recurrent neural networks (RNN).
- DeepAR works well with forecasting timeseries that has seasonality.
- Example: avocado price predictions, stock price forecasting.
- DeepAR can work well in challenging problems when corporates are introducing totally new product to the market with not history (Cold Start Problem).

DEEPAR: OVERVIEW

- DeepAR outperforms classical forecasting techniques such as autoregressive integrated moving average (ARIMA) or exponential smoothing (ETS) in cases where multiple time series forecasting is present.
- DeepAR Allows the training of a single model jointly over multiple related time series (Example: several products demand forecast).
- It could be used to generate point forecast (Number of products to be sold next month is 10,000) and probabilistic forecast (Number of products to be sold next month is between 50,000 and 100,000 with 80% probability).



DEEPAR: USE CASES

PRODUCT DEMAND PLANNING

- DeepAR could be used to forecast inventory levels.
- By feeding in the algorithm with historical of sales, promotions and outlet locations along with weather, website traffic, the algorithm will train a model to generate accuracte product demand forecasts.
- Doing so will empower companies to properly stock inventory in various store locations in anticipation of forecasted demand.

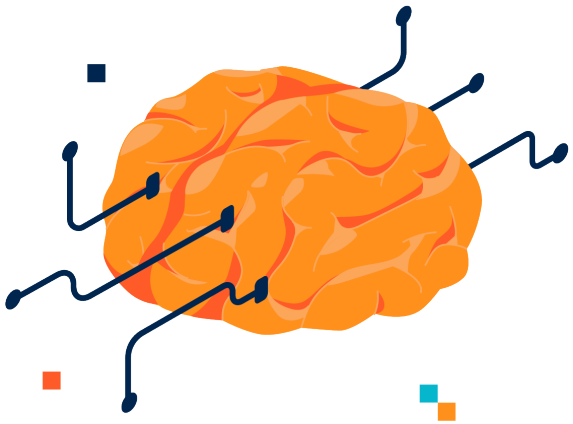
FINANCIAL PLANNING

- DeepAR could be used to accurately predict company's financial information such as revenue, sales, expenses and cash flow.

RESOURCE PLANNING

- DeepAR could be used to make predictions related to number of employees, raw materials, advertising, and revenue.

DEEPAR: INPUT/OUTPUT

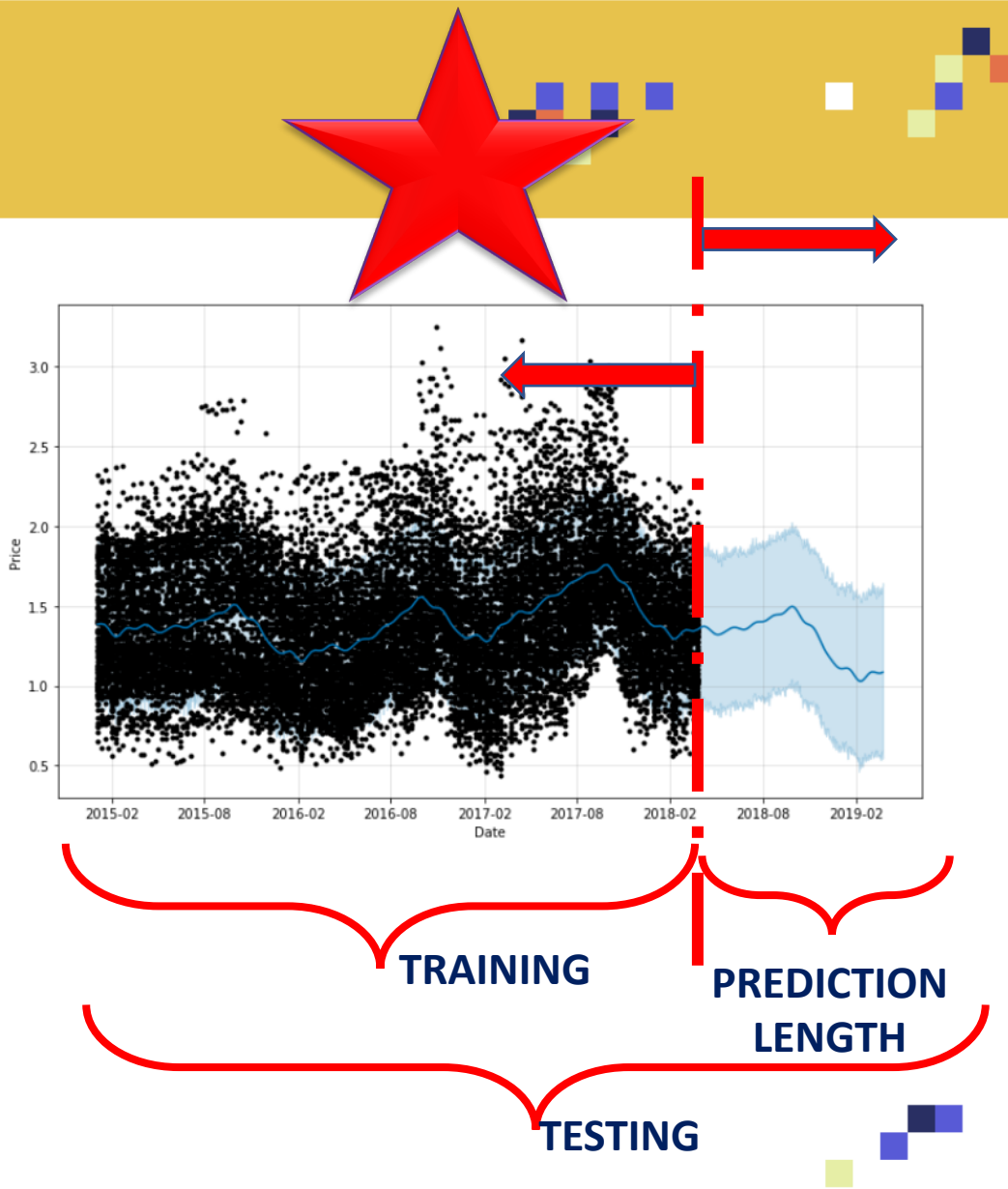


- DeepAR could support two data channels: (1) training and (2) testing
- The testing data channel is optional and could be used for evalution.
- Supported file formats:
  - JSON
  - gzip
  - Parquet
- Data must contain the following:
  - Start: starting time stamp with the format YYYY-MM-DD HH:MM:SS.
  - Target: floating point representing the time series values
  - Dynamic_feat (optional): dynamic features indicates if a promotion was applied to a product or not
  - Cat (optional): if there is categorical features

```
{"start": ..., "target": [1, 5, 10, 2], "dynamic_feat":
[[0, 1, 1, 0]]}
```

DEEPAR: BEST PRACTICES

- When you apply DeepAR, it is recommended that you provide the entire time series for both training and testing.
- Even during model inference, provide the entire time series.
- Do the following:
  - Use the entire dataset as a testing dataset
  - Remove the last prediction_length points from each time series for training.
- Avoid lengthy prediction time (don't use prediction length>400) which will result in large prediction errors.
- DeepAR works well with multiple time series and starts to outperform the ARIMA and ETS.
- For single time series, ARIMA and ETS works better compared to DeepAR



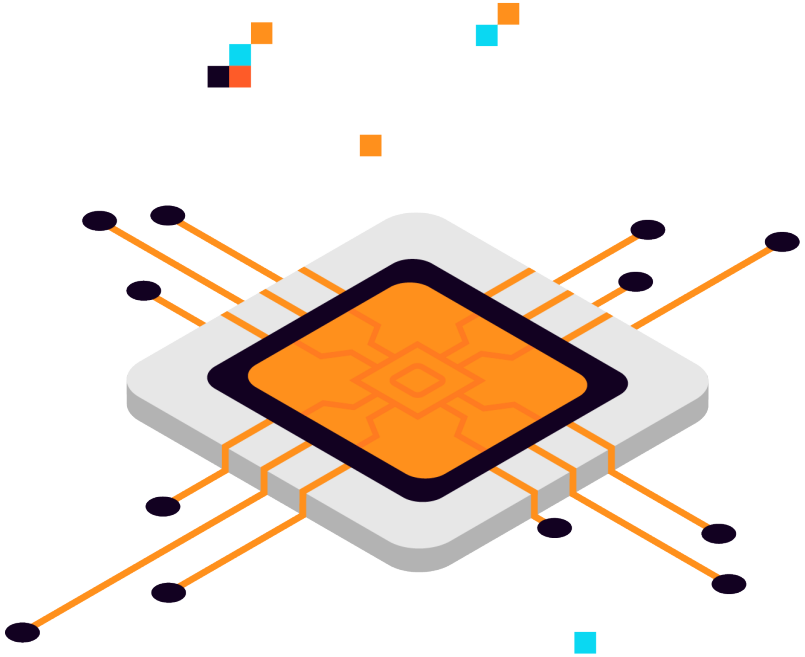
DEEPAR: HYPERPARAMETERS

- Check out the full list of hyperparameters here: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar_hyperparameters.html

- Context_length: The number of time-points that the model gets to see before making the prediction.
- Prediction_length: The number of time-steps that the model is trained to predict
- dropout_rate: The dropout rate to use during training.
- num_dynamic_feat: The number of dynamic_feat provided in the data.
- Epochs
- mini_batch_size
- Learning_rate
- Num_cells

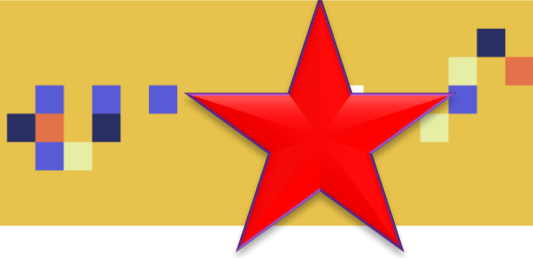
DEEPAR: EC2 INTANCE TYPES FOR DEEPAR

- DeepAR can be trained on both GPU and CPU instances
- DeepAR can be trained in both single and multi-machine settings.
- SageMaker recommends starting out with a single CPU instance (ml.c4.2xlarge or ml.c4.4xlarge)
- Then switch to GPU instances and multiple machines only when:
  - The model is large which occurs when Specifying large values for context_length, prediction_length, num_cells, num_layers, or mini_batch_size
  - During hyperparameters optimization
- DeepAR can only run on CPU instances during inference.



BLAZING TEXT

BLAZINGTEXT: OVERVIEW



- The Amazon SageMaker BlazingText algorithm consists of the following optimized algorithms:

1. Word2vec:

  - Word2vec is used to generate a word embedding
  - Word2vec algorithm is critical in Natural Language Processing (NLP) applications
  - The Word2vec algorithm maps words to distributed vectors
  - Words that are semantically similar correspond to vectors that are close together
  - word embeddings is capable for capturing the semantic relationships between words
  - Skip-gram, batch skip-gram, and continuous bag-of-words (CBOW)
  - Capable of training a model on a billion words in minutes

2. Text classification:

  - Text classification is critical in: web searches, information retrieval, ranking, and document classification.

This is one of the best courses I have seen on Udemy! → **1**

I really hate this course → **0**

BLAZINGTEXT: WORD2VEC DEEPDIVE

- Word2vec was introduced in 2013 and made huge leaps in NLP.
- Since computers only deal with numbers and not text, Word2vec creates a word embedding which is an embedded version of the words that could be fed to a computer.
- Word2vec preserves the relationship between words.
- Word2vec is a shallow simple neural network with a single hidden layer that converts words into vectors

MAN →

0.7
0.08
0.9
2
-4
0.04
0.2
0.5

KING →

0.9
0.09
1
3
-5
0.2
0.4
-0.5

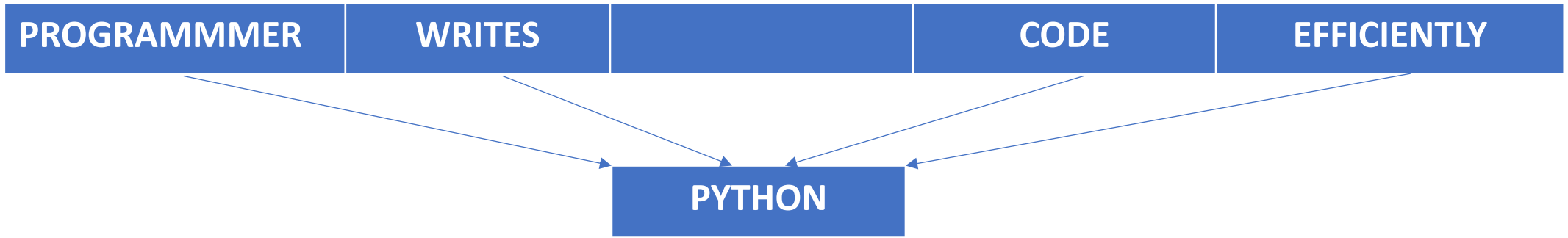
KING-MAN+WOMAN = QUEEN

programmer *writes code in python*
developer *writes code in python*

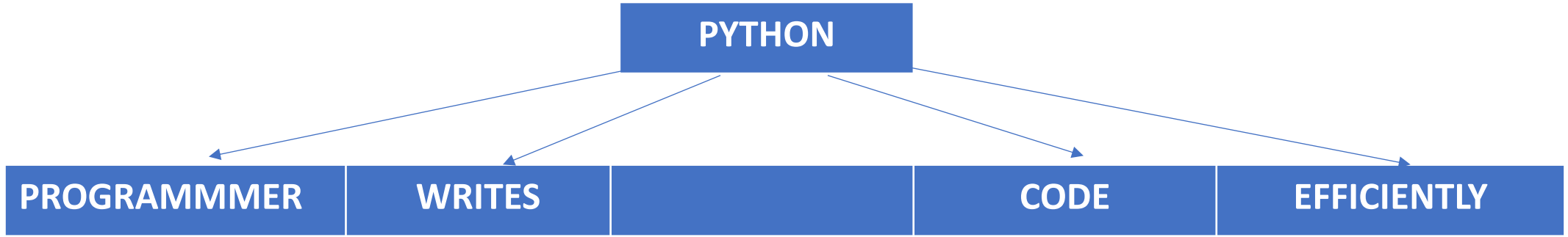
Programmer and developer should have similar embedding vector

BLAZINGTEXT: WORD2VEC DEEPDIVE – CONTINIOUS BAG OF WORDS (CBOW) Vs. SKIMGRAM

- CBOW: predict the target word from the sentence context



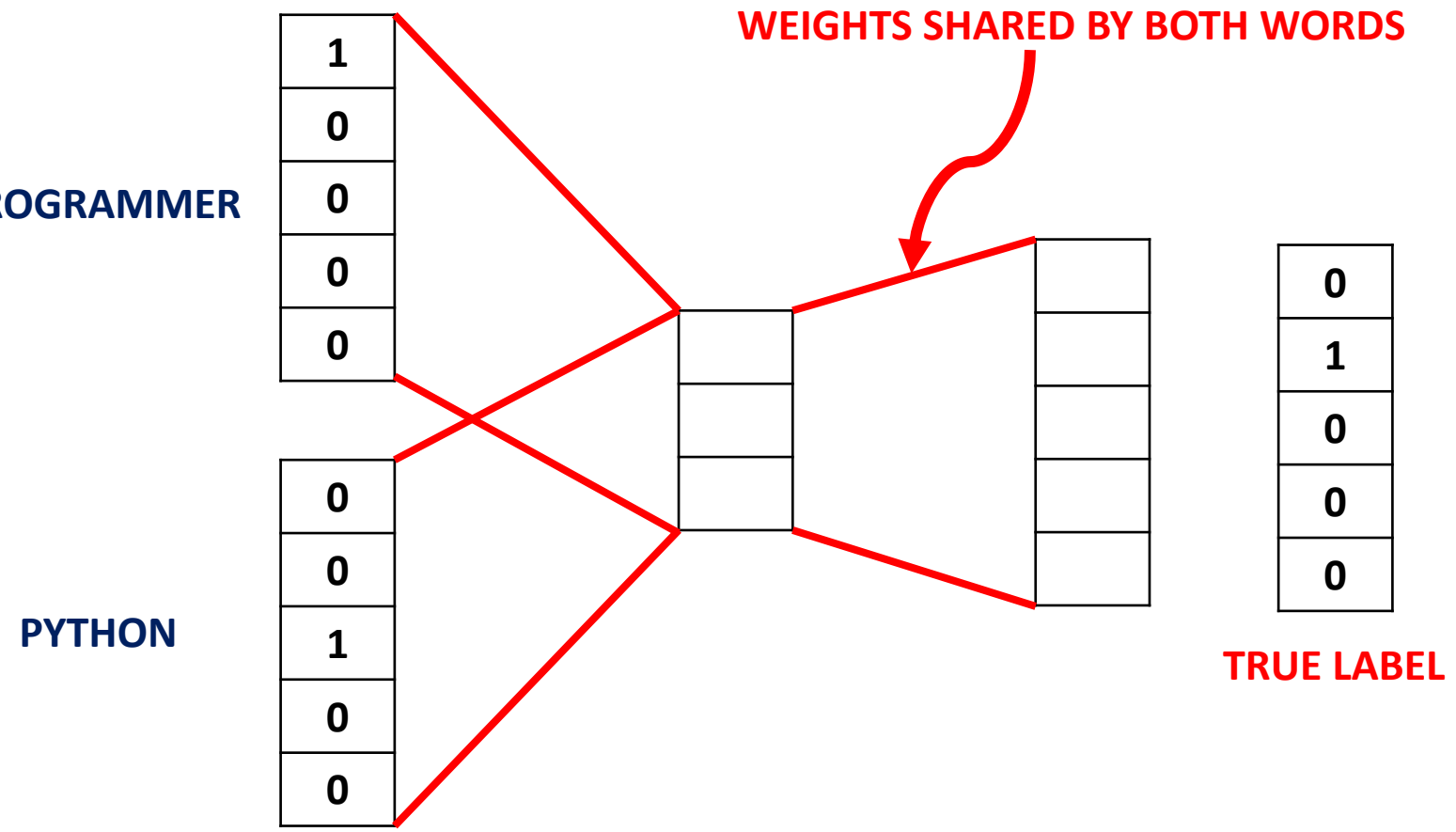
- Skimgram: we try to do the opposite, we try to predict the context from the target word.



BLAZINGTEXT: WORD2VEC DEEPDIVE – CONTINIOUS BAG OF WORDS

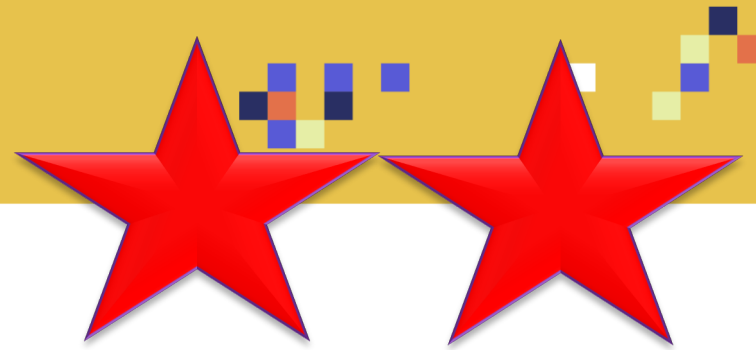
- One hot encoding is performed first.
- Train an artificial neural network where number of inputs represents the total number of words.
- The number of outputs represent the length of the vector.
- The weight matrix represents the set of the vectors.

PROGRAMMER WRITES PYTHON CODE EFFICIENTLY



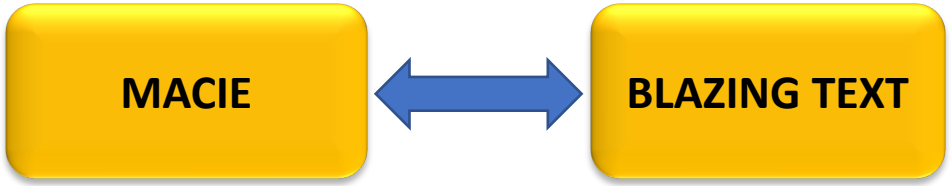
BLAZINGTEXT: USE CASES

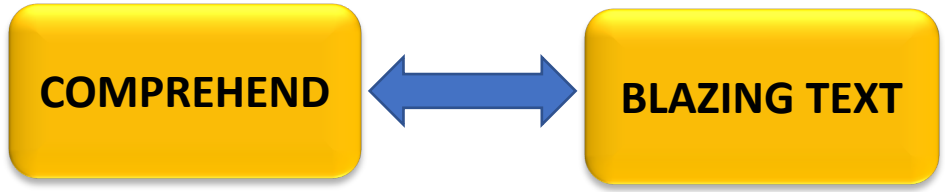
- **Document Classification:**
  - Scan through a corpus of documents and classify them as follows:
    - (1) Contains sensitive data
    - (2) Does not contain sensitive information.
  - Note that Amazon provides a high level service called Macie that could do this.
  - **Amazon Macie** is a security service that automatically discovers and classifies sensitive data.
- **Sentiment Analysis:**
  - Blazing text could be used to perform sentiment analysis by scanning through customer tweets, Facebook posts, and reviews and assess whether customers are happy or not.
  - Note that Amazon provides a high level service called Amazon Comprehend that could provide the same service.



AI/ML HIGH LEVEL SERVICE

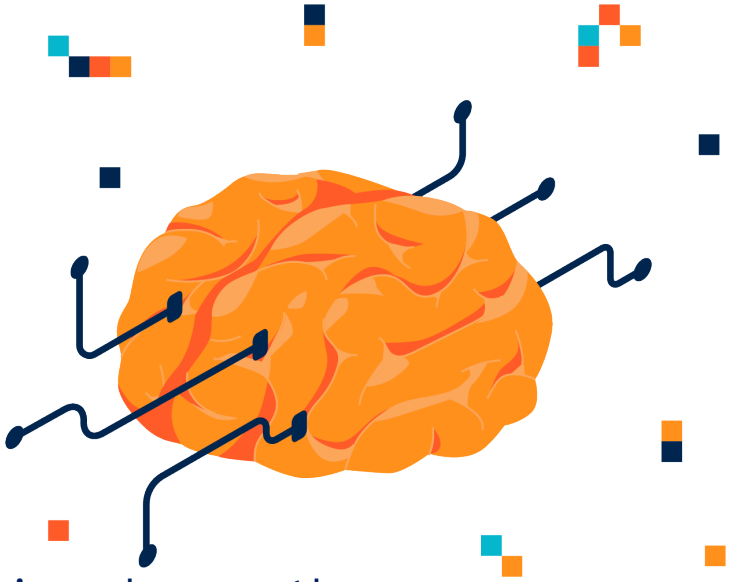
ALGORITHM





BLAZINGTEXT: INPUT/OUTPUT

- The BlazingText algorithm takes in a text file with space-separated tokens.
- In the text file, each line contains a single sentence.
- For supervised mode (text classification):
  - Files should contain a training sentence per line along with the labels.
  - Labels are prefixed by the string __label__.



```
__label__4 linux ready for prime time , intel says , despite all the linux hype , the open-
source movement has yet to make a huge splash in the desktop market . that may be about to
change , thanks to chipmaking giant intel corp .

__label__2 bowled by the slower one again , kolkata , november 14 the past caught up with
sourav ganguly as the indian skippers return to international cricket was short lived .
```

- For unsupervised Blazingtext (Word2vec), it expects a text file with one training sentence per line and no label.

BLAZINGTEXT: HYPERPARAMETERS

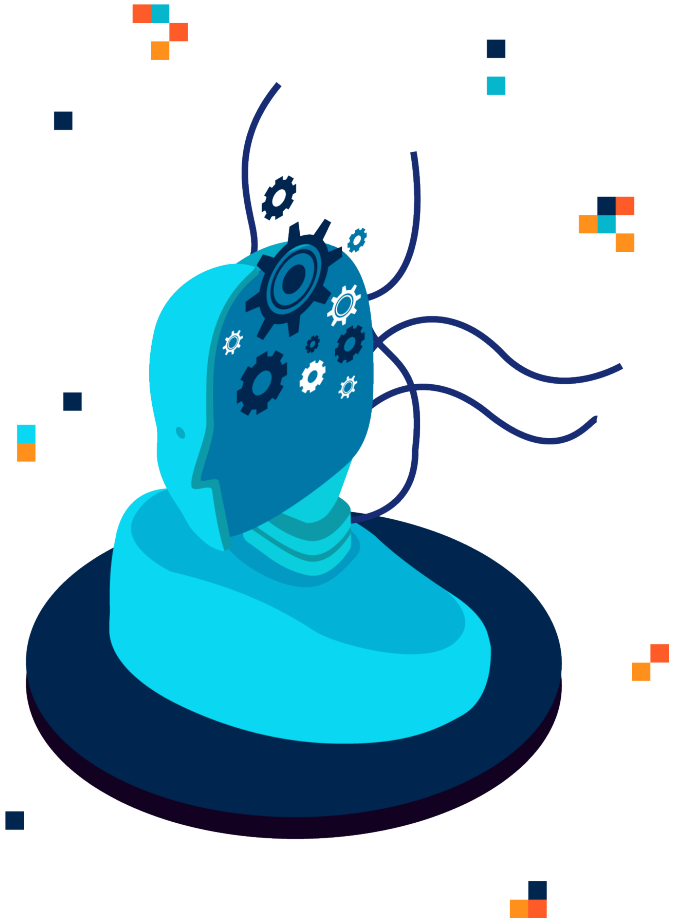
For the full list of hyperparameters, check this out: https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext_hyperparameters.html

Word2Vec Hyperparameters:

- Mode: The Word2vec architecture used for training such as: batch_skipgram, skipgram, or cbow
- batch_size: The size of each batch when mode is set to batch_skipgram.
- Epochs
- Learning rate

Text Classification Hyperparameters:

- Mode: The training mode, valid values: supervised
- early_stopping: Whether to stop training if validation accuracy doesn't improve after a patience number of epochs.
- Epochs
- Learning rate
- vector_dim: The dimension of the embedding layer
- word_ngrams: The number of word n-gram features to use.



BLAZINGTEXT: EC2 INSTANCES

- For cbow and skipgram modes, BlazingText supports single CPU and single GPU instances.
- ml.p3.2xlarge instance is recommended.
- For batch_skipgram mode, BlazingText supports single or multiple CPU instances.
- For supervised text classification, C5 instance is recommended if the training dataset is less than 2 GB.
- For larger datasets, use single GPU (ml.p2.xlarge or ml.p3.2xlarge).