Image Optimization Model

RealTimeML Predictive Analytics for Email Engagement

By Miu Lun (Andy) Lau

loxz digital

Table of Contents

©2022 All Rights Reserved www. loxz.com

Abstract

The goal of this ML model is to provide RealTime Predictive Analytics on the recommended images to provide accurate email engagement metrics and timely feedback for email related marketing campaigns. We utilizes an automotive image dataset and generate additional images using data augmentation technique and pipeline. This particular prediction model utilizes the **CNN image based ResNet model,** and we performed transfer learning using the existing parameters and weights from an ImageNet dataset. The resultant model is then further improved using XGboost classification algorithms to increase its accuracy. The training and deployment is conducted on AWS Sagemaker. Our algorithms obtained an **accuracy score of 91%** using the testing dataset.

I . INTRODUCTION

The demonstration of this model is to provide RealTime predictive analytics on the images used for digital marketing campaigns. This model will identify the best engagement images for the target variable chosen and provide potential or recommended images with a higher engagement rate. For demonstration purposes, this model was trained specifically using an automotive related dataset, but can be tuned for other type of imagery depending on the dataset methods.

Ⅱ. Datasets and Features

The dataset used in this analysis is from the **Stanford AI Lab Repository[4].** It contains a total number of 16,185 images of 196 classes of cars. From there, we performed data analytics to generate synthetic data using data wrangling processes. We generated synthetic data that would be typical for an email digital marketing campaign, which will be used to train the machine learning model. Further these images can be changed by using various data augmentation techniques to increase the size of the dataset for additional accuracy and performance. Using the model inference, we can derive in real time whether an augmented image would result in a higher engagement rates.

Future models will utilize existing data with the ability to continuously introduce new data to the desired target variables; however, for this iteration we synthetically created variables and assigned random conversion rates.

These are our current **implemented target features** and **future target features.** For the images, we introduce an image augmentations pipeline using Albumentations [1].

- **Open Rate**
- **Click Through Open Rate (CTOR)**
- **Revenue Generate Per Email**
- **Conversion Rate**
- **Purchase Rate**
- **Unsubscribe Rate**
- **Bounce rate**

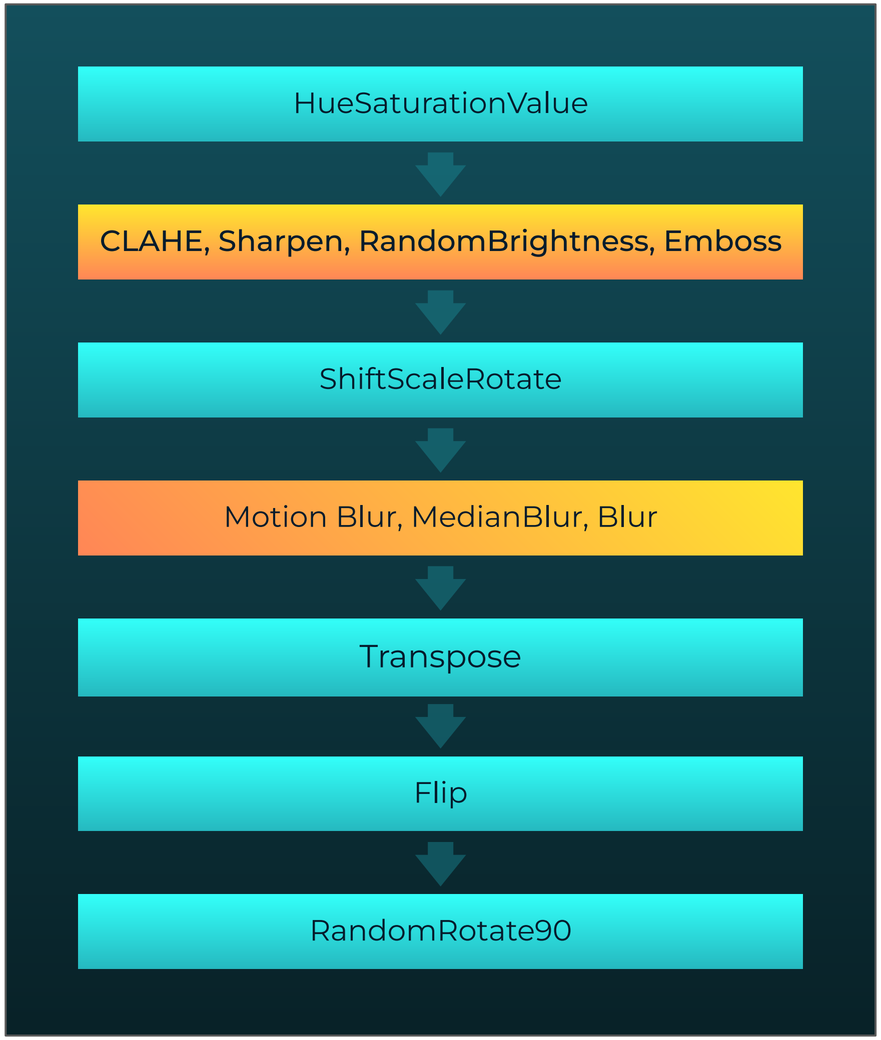

HueSaturationValue
CLAHE, Sharpen, RandomBrightness, Emboss
ShiftScaleRotate
Motion Blur, MedianBlur, Blur
Transpose
Flip
RandomRotate90

Figure 1. Image Augmentation Pipeline

A. Data Engineering and Wangling

The goal is to create and implement a synthetic dataset for introducing email related campaign models. We utilize *Albumentations*[1] to construct an image augmentation pipeline. The augmentation pipeline can be seen in figure 1.

The augmentation pipeline consists of a number of blocks where each contains a type of image augmentation technique. In each of the blocks, the images will have a probability ***p*** of applying the such features. The first couple augmentations contain common translation and rotation such as *Flip*, *GaussNoise*, *Transpose* etc. The fourth block will randomly select from one of the three augmentation techniques (*MotionBlur*, *MedianBlur*, and *Blur*). The next block will show *Shiftscalerotate* which randomly applies affine transform of (translate, scale, and rotate). The subsequent block will select from one of (*CLAHE*, *Sharpen*, *Emboss*, and *RandomBrightness*). Finally, the last block will be modified to the *HueSaturationValue*. A list of resulting augmented images can be observed in figure 2.

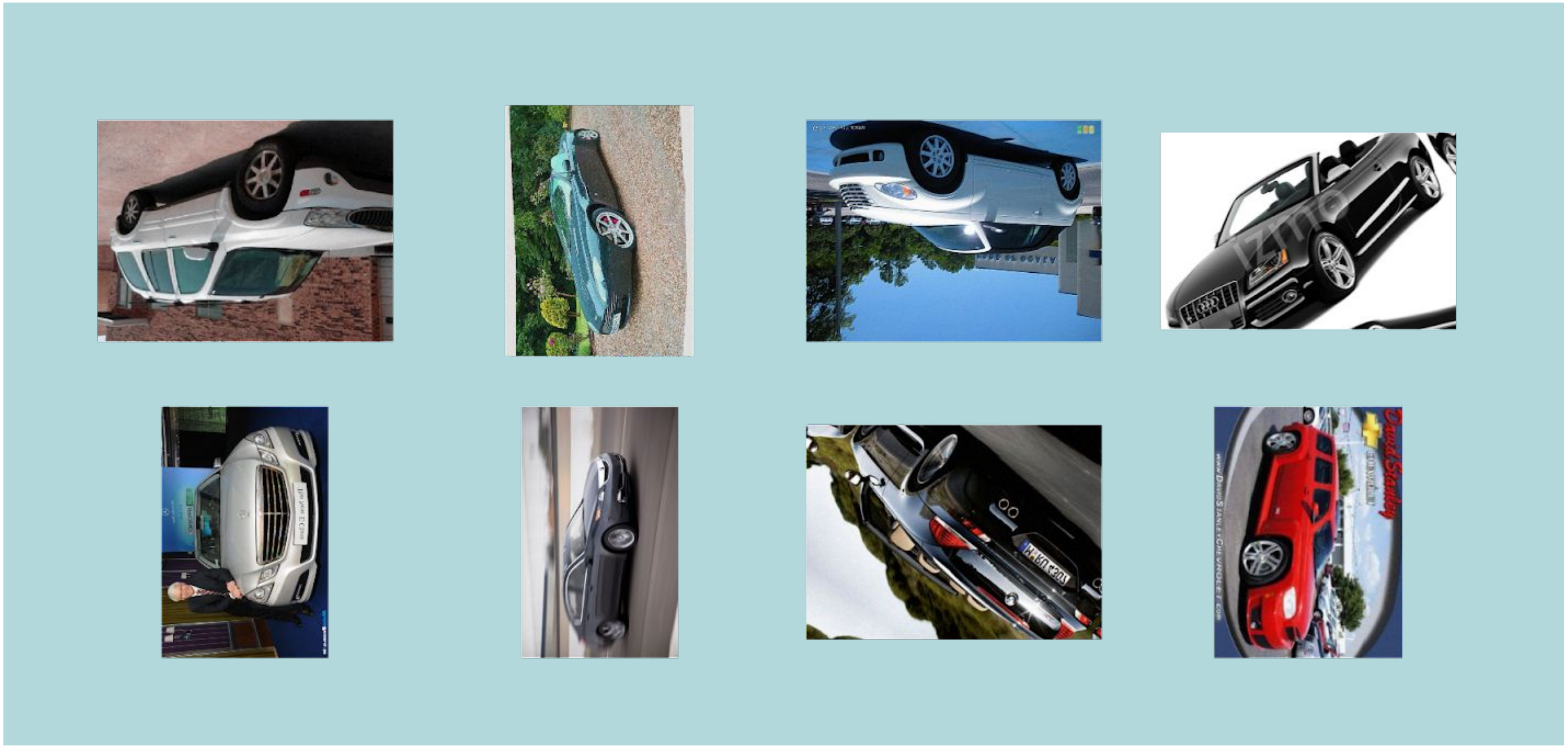


Figure 2. A sample of eight images which result from the image augmentation pipeline.

B. Feature Engineering

To simulate real world email conversions, our synthetic variables, the target features, were randomly assigned a binary success variable based on parameters. There are a total number of three target variables. For the target variables *Open Rate* and *Click Through Open Rate*, we assigned 80% to be a successful conversion if the image was a brighter augmentation or a rotated augmentation, whereas the other augmentations were assigned 20% to be a successful conversion. This percentage assignment is arbitrary and is only meant to highlight the potential effectiveness of our model. The current metadata associated with our synthetic data is shown below in table I.

Metadata	Meaning
Relative_im_path	Relative file path to the actual image
Class_name	Full description of the vehicle
Make	Make of the vehicle
Vehicle_type	Type of vehicle
Year	Year of vehicle
Open Rate	Binary value for a open rate event
Revenue_Per_Email	Binary value for a revenue generated event
Click_to_Open_Rate	Binary value for a click-through event
Conversion_Rate	Binary value for a conversion event
Campaign types	[Abandoned Cart, Newsletter, Promotional, Survey,Transactional, Webinar, Engagement, Review_Request, Product Announcement]
Industry types	[Automotive, Industrial, Real Estate, Transportation and Logistics, Finance and Banking, Professional and Business Services, Manufacturing, Film and Motion Picture, Retail,Software and Technology, Hospitality, Healthcare, Academic and Education]

Table 1. Data-set meta data and the corresponding value for campaign type and industry type. (https://chart-studio.plotly.com/~laumiulun/48/#plot)

Ⅲ. TOOLS REQUIRED

We are using a number of software and packages frequently used by the data science community. For machine learning, we are using **Pytorch** [5]. The image augmentation is done using **Albumentations.** Various other commonly used packages are employed for data parsing and visualizations such as **Numpy, Matplotlib, Pandas,** and **Scipy**. PIL is used for Image processing and IO, and Ipywidgets is used for demonstration and selection of targets and other parameters. BOTO3 is used for accessing and interacting with S3 data storage.

IV. MODEL ALGORITHMS

To implement the email related features, we extracted image embeddings using a pre-trained Convolutional Neural Network (CNN). The specific model is implemented using ResNet, which is a popular image classification algorithm. There are many versions of ResNet with different numbers of layers. We will be focusing on two specific ResNet models with 18 and 34 layers. Both models contain pretrained weights from ImageNet dataset [3]. We then used these image embeddings as further inputs into a Gradient Boosted Tree model to generate probabilities on a user specified target variable. For higher accuracy scores, we can use ensemble methods. Future versions of this model will have accuracy scores to better explain the model.

- Convolutional Neural Network using ResNet models:
  - ResNet with 18 layers
  - ResNet with 34 layers
    - Gradient Boosted Tree Model using XGBoost

V. MODEL DEVELOPMENT

The first step of our model utilizes the pretrained ResNet34 CNN. CNNs are neural networks most commonly applied to visual imagery analysis and image recognition, we use a pre-trained version with 34 hidden layers to better vectorize the images embedding – where vector lengths equal to the number of augment classes. We utilize the **Pytorch ML** framework for this since it contains powerful pretrain algorithms and easy implemented techniques. The use of this algorithm correctly recognizes images to their augmented labels which is significant for the next step of our model. To train and test, the dataset was split before training is initialized and maintained as a ratio of 20%. Our training process utilizes a total number of **10 epochs**. The accuracy of the model by using the testing dataset at the end of each epoch. The criterion is designated as *CrossEntropyLoss* which calculates the cross entropy loss between input and target, and the reduction method is set as “mean". The optimizer is set as stochastic gradient descent. To further optimize the model, a learning rate scheduler is implemented which reduces the learning rate by a factor once learning stagnates. Our **learning rate scheduler (LRS)** is set using *ReduceLROnPlateau* which reads a metrics quantity and if no significant improvement is seen for a ‘patience’ number of epochs, the learning rate is then reduced. The **LRS** is set using the ‘max’ mode, and the patience is set as three epochs, where one epoch represents one cycle through the whole dataset. Machine learning models require many computational hours in-order to retrain using the new data obtained, the XGBoost[2] algorithm allows for rapid training while retaining some of the best predictions power as shown below in Figure 4. We can observe that XGBoost resulted in similar accuracy compared to gradient boost while requiring a factor less in training time. Once we obtain our vectorized image embeddings, we feed them to our XGBoost Classification algorithm. This ensemble algorithm has fast execution speeds and hosts optimal model performance amongst other methods in classifying the success of a target variable from these vectors.

To further prepare our model for development, we utilize the AWS Sagemaker for training and deployment. The training and hyper-parameter optimization procedure will be conducted in AWS Sagemaker using Notebook Instances. The training data will be hosted on S3 Bucket which keeps track of the training jobs and as well log files.

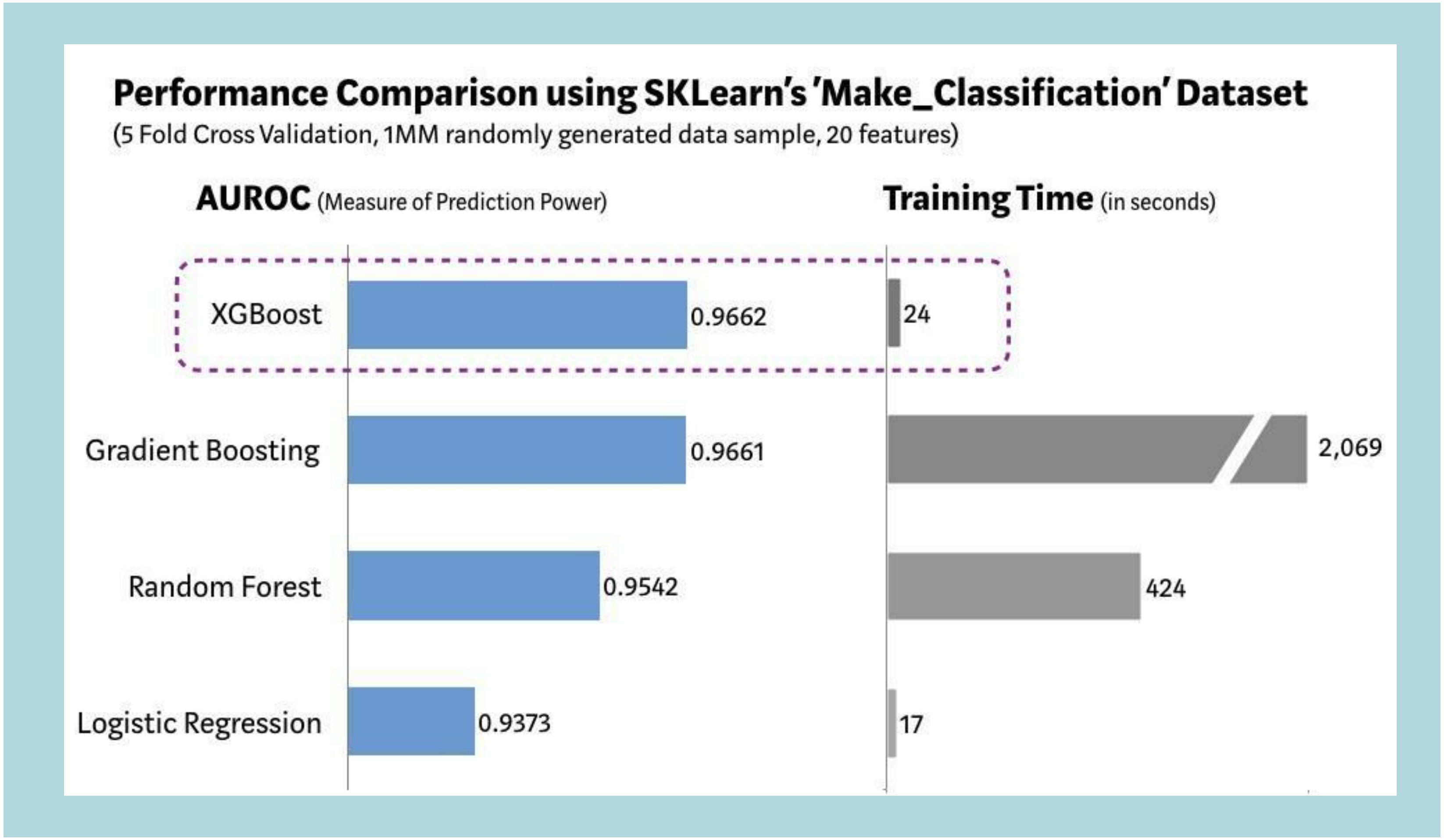


Figure 3. Comparisons of XGBoost and various other optimizations techniques. Visual and data obtained from Vishal Morde [8].

VI. AWS SETUP

AWS Sagemaker contains a variety of different configurations for both training and inferences, we wish to test various configurations to optimize the training procedure. There are a total number of 16 different instances in AWS, with 7 being CPU (Compute optimized) and 9 being GPU (Accelerated Computing). The training script is designed to make use of either CPU or GPU accelerated workflow. Dataset is distributed across using AWS Sagemaker SDK related commands. For GPU accelerated workload, the model and data is sent to the GPU before training is to proceed. An example of AWS instances and the corresponding training time is shown below in table II. The hyper-parameters are identical and contain the same dataset located in an S3 bucket. We can observe the significant speedup obtained from using GPU accelerated instances, where the V100 GPU instances are significantly faster than T4 GPU. Meanwhile, the CPU instances are not observed to receive similar speedup with even worse performance with an increased number of CPUs.

VII. MODEL ASSUMPTIONS AND VALIDATION

The models are built in Amazon SageMaker, where it contains a number of models pretrained from their model zoo. We first obtained the models (ResNet18 and ResNet34) and retrained our models using our custom automotive related dataset. The dataset currently is stored locally in SageMaker, but can also be imported directly from Amazon S3 Buckets data storage to allow for real time optimizations.

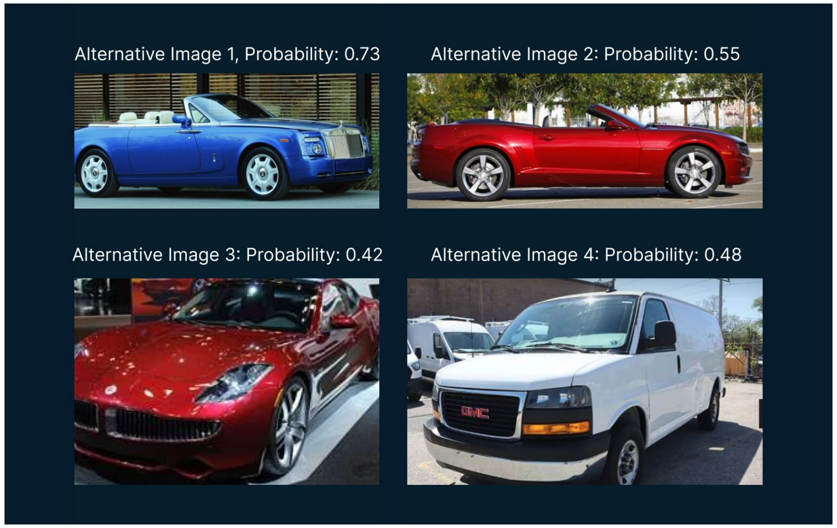

Alternative Image 1, Probability: 0.73
Alternative Image 2: Probability: 0.55
Alternative Image 3: Probability: 0.42
Alternative Image 4: Probability: 0.48

Figure 4. Example of image suggestions with their corresponding probability.

VIII. MODEL TUNING AND PERFORMANCE

The selection of hyper-parameters is very important in-order to increase the classification metrics, these include learning rate, and weight decay. Typical methods includes gridsearch, random, or sobol. However, we utilize the powerful automatic model tuning technique in Amazon SageMaker SDK [6, 7], which utilizes a Bayesian related search technique to discover the optimal hyper-parameters set. Sagemaker allows for various selection when dealing with hyper-parameters optimizations, such as *ContinousParameters*, *IntegerParameters*, and *CategoricalParameters*. Afterward, we utilizes regex to extract the metrics from AWS Cloudwatch and compute the optimal hyper-parameters.

IX. DATA SCIENTISTS OWN CONCLUSIONS

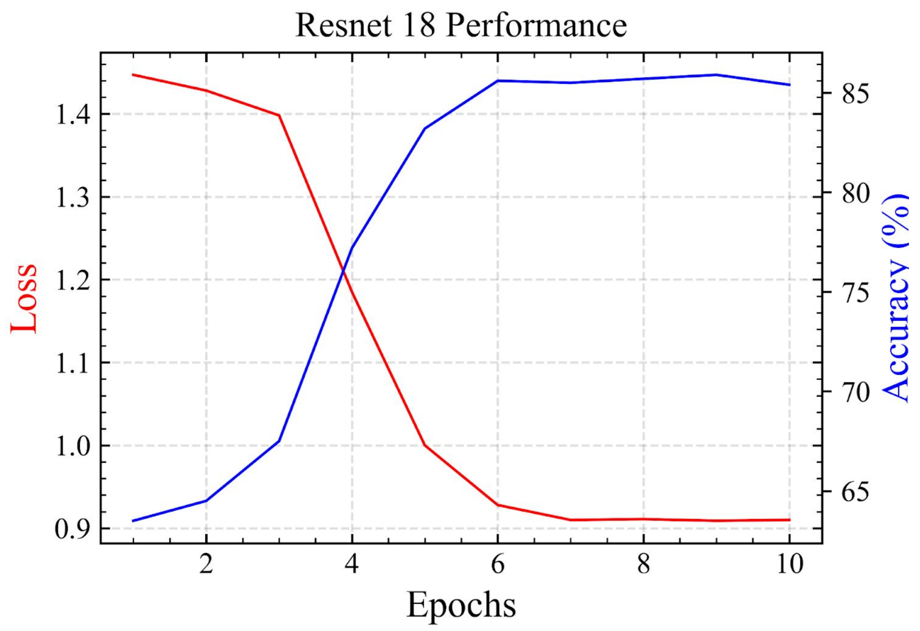
We can observe the performance of the model is performing well on our dataset, which validates the correct usage of both the dataset and model implementations. The models demonstrate good accuracy in predictions and recommendations for alternative suggestions with higher probabilities. The models can be used to predict various target variables depending on the campaign's desires. The accuracy of ResNet18 achieved 86% after 10 epochs, and ResNet34 achieved 91%. We can observe that ResNet34 performs better but ResNet 18 also perform reasonable well given it much smaller layer depth.There are trade offs in both models in that ResNet 18 provides fast training and inference time but lower in accuracy, ResNet 34 provides higher accuracy but can be much slower in training time. The decision will ultimately be made by the End user.

Instance Type	Num CPU	Mem (GBs)	Num GPU	Training (hrs)
m1.c5.large	2	4	0	4.64
m1.c5.xlarge	4	8	0	4.11
m1.c5.2xlarge	8	16	0	3.94
m1.c5.4xlarge	16	32	0	3.82
m1.c5.9xlarge	36	72	0	3.67
m1.c5.12xlarge	48	96	0	3.87
m1.c5.18xlarge	72	144	0	3.95
m1.p3.2xlarge	8	61	1 V100	0.56
m1.p3.2xlarge	32	244	4 V100	0.43
m1.p3.2xlarge	64	488	8 V100	0.37
m1.g4dn.xlarge	4	16	1 T4	1.42
m1.g4dn.2xlarge	8	32	1 T4	1.65
m1.g4dn.4xlarge	16	64	1 T4	1.34
m1.g4dn.8xlarge	32	128	1 T4	1.55
m1.g4dn.12xlarge	64	256	1 T4	1.75
m1.g4dn.16xlarge	48	192	4 T4	1.09

Table 2. Testing AWS Sagemaker instance types on the training speed. All models were tested with identical variables and hyper-parameters. (https://chart-studio.plotly.com/~laumiulun/51/)

X. CONCLUSION AND FUTURE CONSIDERATIONS

We constructed a model for synthetic email campaign dataset for real time optimizations. The model utilizes an labeled automotive dataset which contains various meta data such as vehicle make and type. The amount of image were doubled using image augmentation techniques to increase the amount of training data available. We utilizes SageMaker to train and deploy the model for inferences. Both models demonstrates good accuracy in predictions and recommendations. The two ResNet model candidate is suitable for the next step in further optimizations in both hyper-parameter and metrics modifications. Additional other ResNet models can also be used as a future study point.



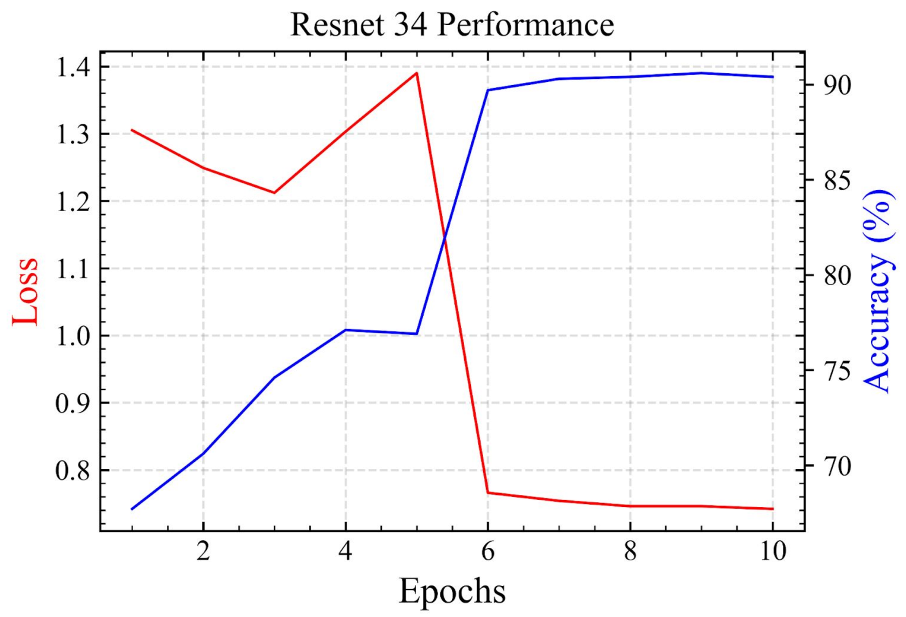


Fig. 5: Example of loss and accuracy in ResNet18 (https://chart-studio.plotly.com/~laumiulun/46/#/) and ResNet34 (https://chart-studio.plotly.com/~laumiulun/44/#/)

References

[1] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: Information 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https:// www.mdpi.com/2078-2489/11/2/125.

[2] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2016, pp. 785–794.

[3] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: 2009 IEEE conference on computer vision and pattern recognition. Ieee. 2009, pp. 248–255.

[4] Jonathan Krause et al. "3d object representations for fine-grained categorization". In: Proceedings of the IEEE international conference on computer vision workshops. 2013, pp. 554–561.

[5] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[6] Valerio Perrone et al. "Amazon Sage-Maker automatic model tuning: Scalable black-box optimization". In: arXiv preprint arXiv:2012.08489 (2020).

[7] Valerio Perrone et al. "Amazon sagemaker automatic model tuning: Scalable gradient-free optimization". In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021, pp. 3463–3471.

[8] Morde Vishal. Tree-Based Machine Learning Algorithms Explained. Ed. by Medium.com. Medium.com [Online; posted 19-June-2021]. June 2021. 6