Convolutional Neural Network Based Intrusion Detection System

CSE534-Project-Progress Report, Fall 2019

Instructor: Aruna Balasubramanian

TA: Javad Nejati

Student: Jiaxiang Ren (112748304)

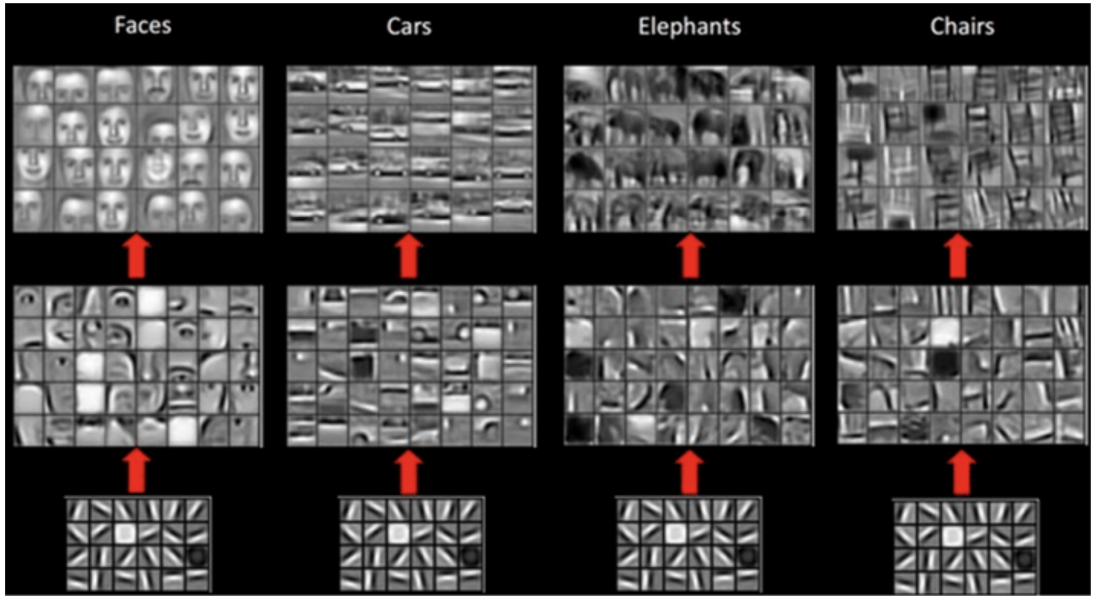
Contents

Introduction - Problem Statement

- Intrusion Detection (ID)
  - Identify network traffic
  - A classification task, binary (normal/attack) or multi-class (normal and different kinds of attack)

- Machine learning approaches
  - Widely used in ID and achieved relative high performances
  - **Feature extraction**, classification, validation
    - Crucial issue: the robust features requiring considerable expertise, lots of time and resources

Introduction - Approach



- Deep learning approaches
  - Automatically feature extraction
  - Powerful representation for inference
  - Convolutional Neural Networks (CNNs) extract features hierarchically
  - Even outperform classic machine learning methods
- Some points not mentioned:
  1. How to design the header (classifier)
     - CNN Model = CNN (feature extractor) + **header (classifier)**
  2. Some state-of-the-art techniques have not been tested yet
     - Improvement space
  3. The robustness

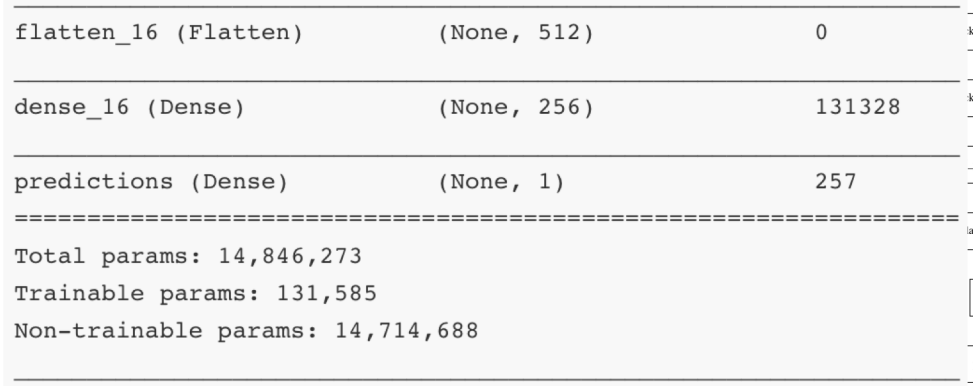
Solution (contribution)

- Design the header (classifier)
  1. Prototype Model
  2. Model with Modified Header
- Light Weight Model
- Data Augmentation

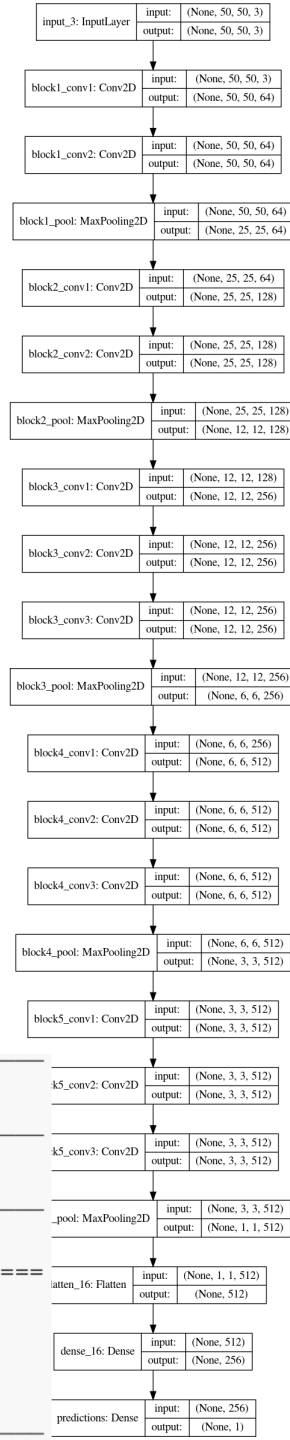
Solution - Design the header

- Prototype Model
  - Feature extractor: VGG16
  - Header: 3 Fully Connected Layers → 2 small Fully Connected Layers
- Design idea:
  - Reduce header size as many as possible
  - Less parameters, less overfitting risk
  - 131,585 trainable parameters in total
  - Reduces the classifier size into 1/175

$$512 * 4096 + 4096 * 4096 + 4096 * 1000 = 22970368$$
$$512 * 256 + 256 * 1 = 131328$$
$$22970368/131328. \approx 175$$



```
flatten_16 (Flatten)         (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               131328
_________________________________________________________________
predictions (Dense)          (None, 1)                 257
=================================================================
Total params: 14,846,273
Trainable params: 131,585
Non-trainable params: 14,714,688
_________________________________________________________________
```

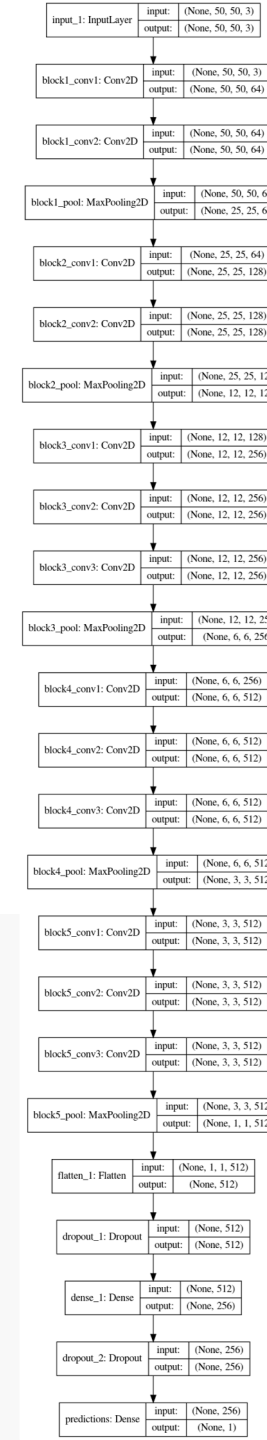


Solution - Design the header

- Model with Modified Header
  - Feature extractor: VGG16
  - Header: 3 Fully Connected Layers → 2 Fully Connected Layers with Dropout, BatchNormalization and Parametric Rectified Linear Unit
- Design idea:
  - Small header size
  - 131,585 trainable parameters
  - Adding widely used techniques

flatten_4 (Flatten)	(None, 512)	0
dense_3 (Dense)	(None, 256)	131328
dropout_2 (Dropout)	(None, 256)	0
predictions (Dense)	(None, 1)	257

Total params: 14,846,273
Trainable params: 131,585
Non-trainable params: 14,714,688



Solution – Light Weight Model

- Light Weight Model
  - Feature extractor: Stack[illegible] of Conv Blocks
    (with dilated Conv Laye[illegible]
  - Header: 2 Fully Connec[illegible]
- Design idea:
  - Reduce model size as m[illegible]
  - Adding widely used tec[illegible]
  - The total trainable parameters are **49,381** which are fa[illegible] previous two models (only ~**37%**)

global_average_pooling2d_1 (	(None, 48)	0
dropout_1 (Dropout)	(None, 48)	0
dense_1 (Dense)	(None, 256)	12544
p_re_lu_1 (PReLU)	(None, 256)	256
dropout_2 (Dropout)	(None, 256)	0
predictions (Dense)	(None, 1)	257

Total params: 49,669
Trainable params: 49,381
Non-trainable params: 288

Header

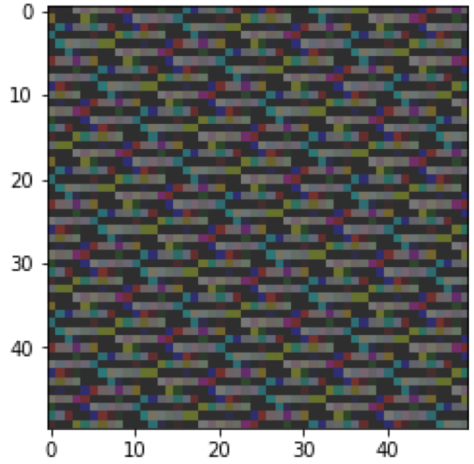
Feature extractor

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d_1 (Conv2D)	(None, 32, 32, 12)	912
batch_normalization_1 (Batch	(None, 32, 32, 12)	48
activation_1 (Activation)	(None, 32, 32, 12)	0
conv2d_2 (Conv2D)	(None, 32, 32, 12)	1308
batch_normalization_2 (Batch	(None, 32, 32, 12)	48
activation_2 (Activation)	(None, 32, 32, 12)	0
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 12)	0
conv2d_3 (Conv2D)	(None, 16, 16, 24)	2616
batch_normalization_3 (Batch	(None, 16, 16, 24)	96
activation_3 (Activation)	(None, 16, 16, 24)	0
conv2d_4 (Conv2D)	(None, 8, 8, 48)	10416
batch_normalization_4 (Batch	(None, 8, 8, 48)	192
activation_4 (Activation)	(None, 8, 8, 48)	0
conv2d_5 (Conv2D)	(None, 8, 8, 48)	20784
batch_normalization_5 (Batch	(None, 8, 8, 48)	192
activation_5 (Activation)	(None, 8, 8, 48)	0

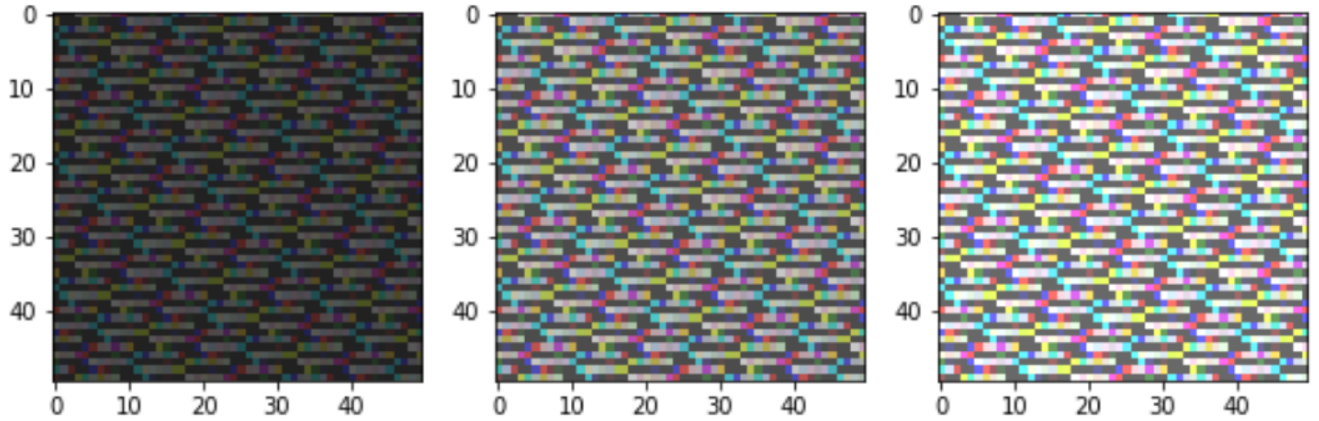
Solution - Data Augmentation

- Data Augmentation
  - A series of operations to transform training sample into another similar but different feature
    - add noises
    - random crop data
  - Make model more generalized for variant changes in data
  - Expand training samples
    - especially for small and deficient dataset
    - apply at the beginning of each training iteration

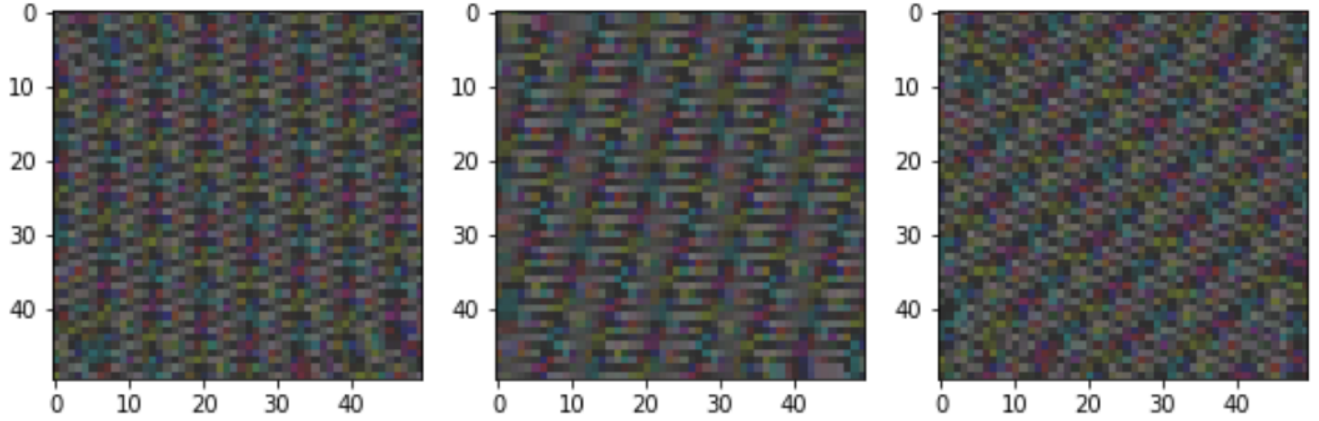
Solution - Data Augmentation



1. Random value shift: Add integer value randomly on each byte.

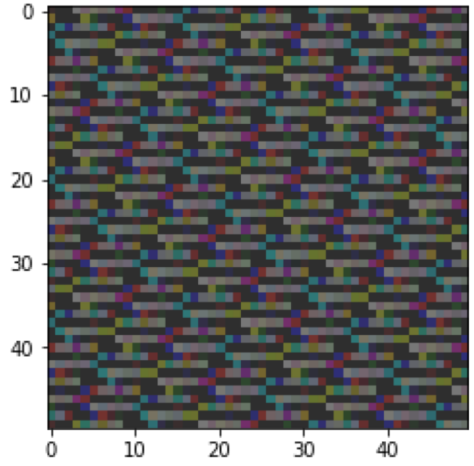


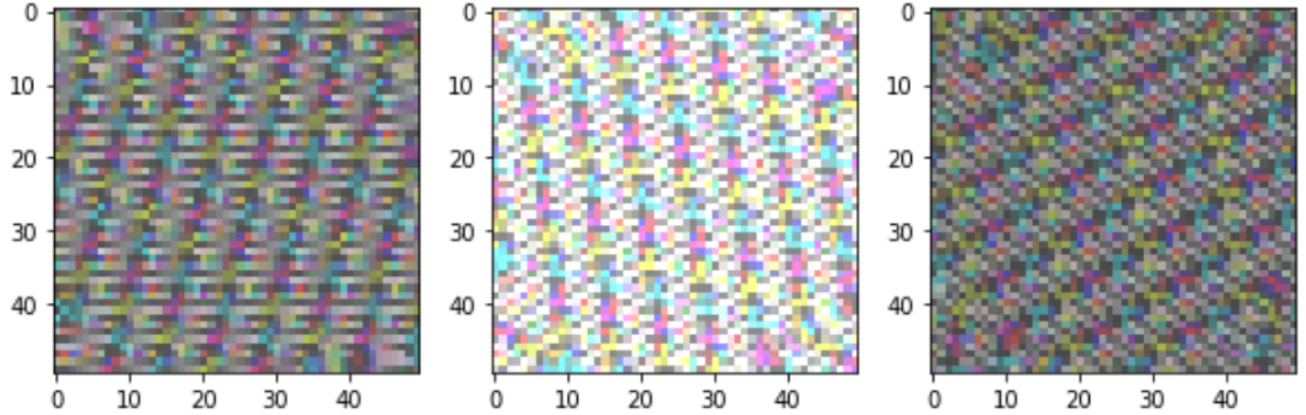
2. Data rearrangement: Rearrange the order of bytes in a package. For this project, this operation is implemented as rotation of visualized data payload.



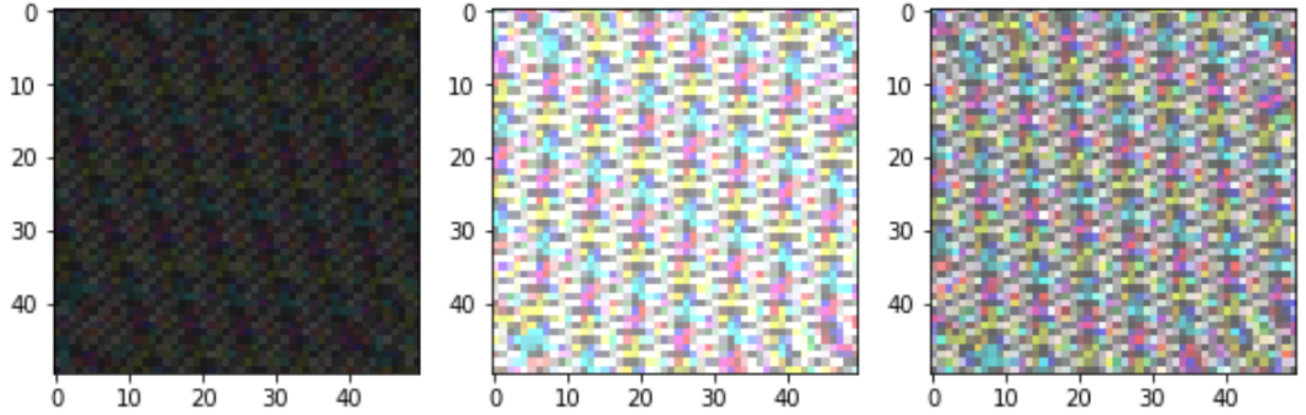
Solution - Data Augmentation

And the combination of the above two approaches:

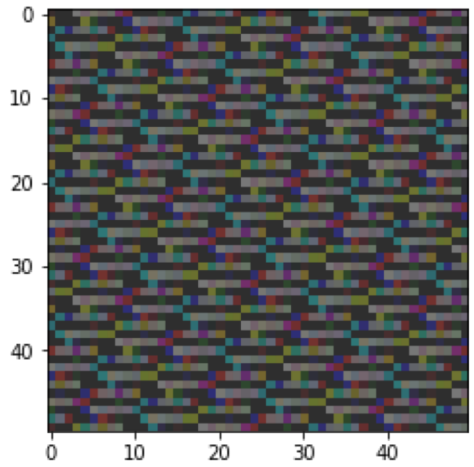


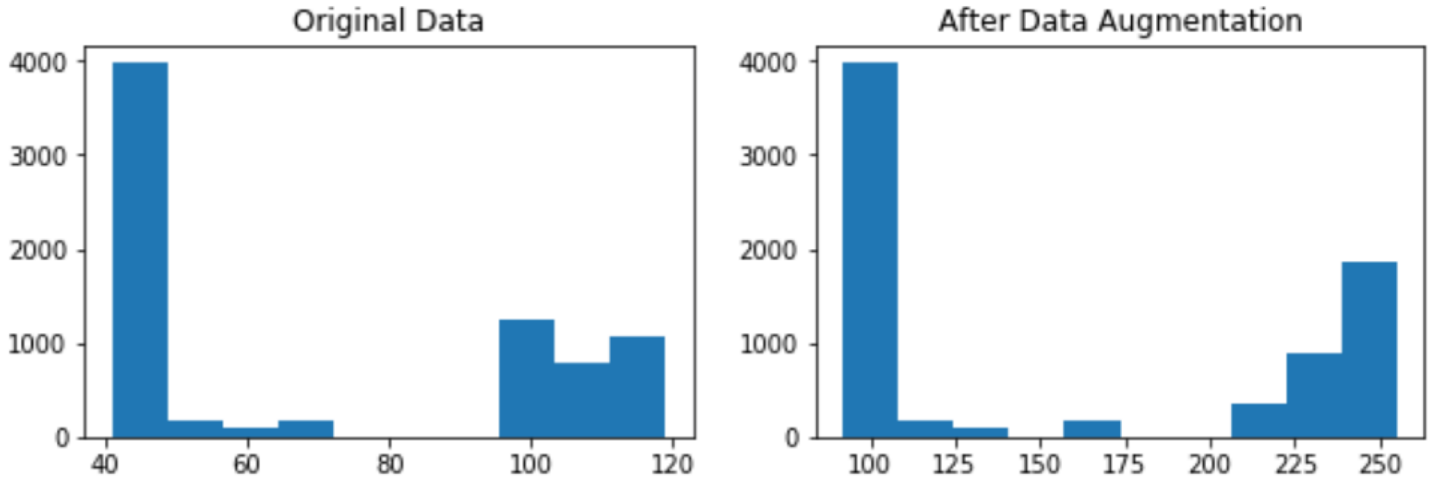


Random value shift + Data rearrangement



Solution - Data Augmentation





Histograms of the original data and the data after data augmentation.

Evaluation - Metrics

- Evaluation metrics:
  - accuracy
  - precision
  - recall rate
  - Plot Receiver Operating Characteristic (ROC) curve and calculate Area Under Curve (AUC)

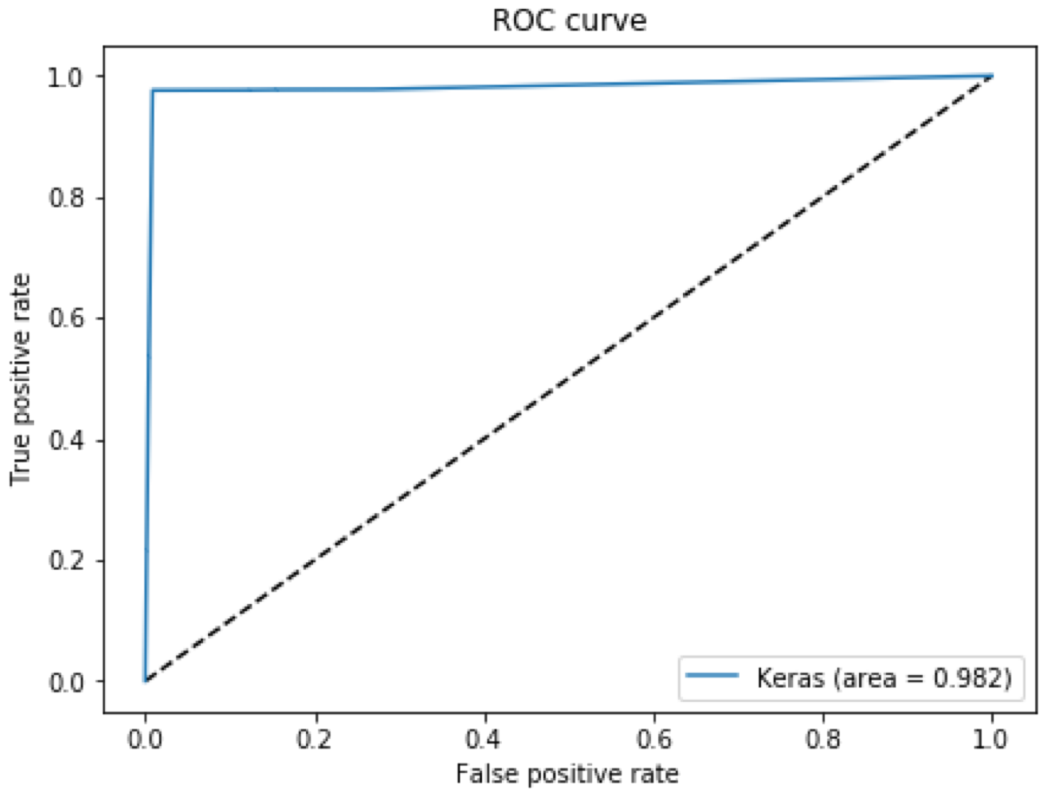
Result

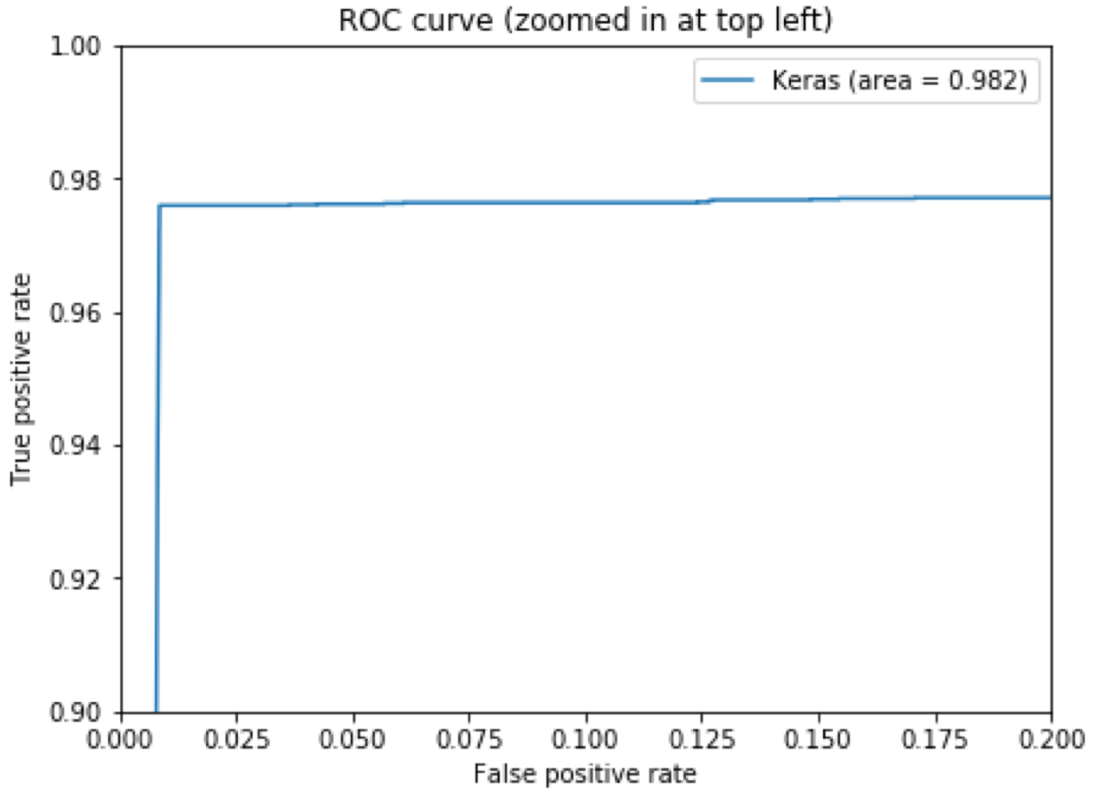
All the data are split randomly into 80% training set and 20% validation set.

	Accuracy	**AUC**	**Time Cost (s/epoch)**
Prototype Model	**0.99066**	0.982	167
Model with Modified Header	0.99064	0.983	176
Light Weight Model	0.99056	**0.993**	**67**

Result
Prototype Model

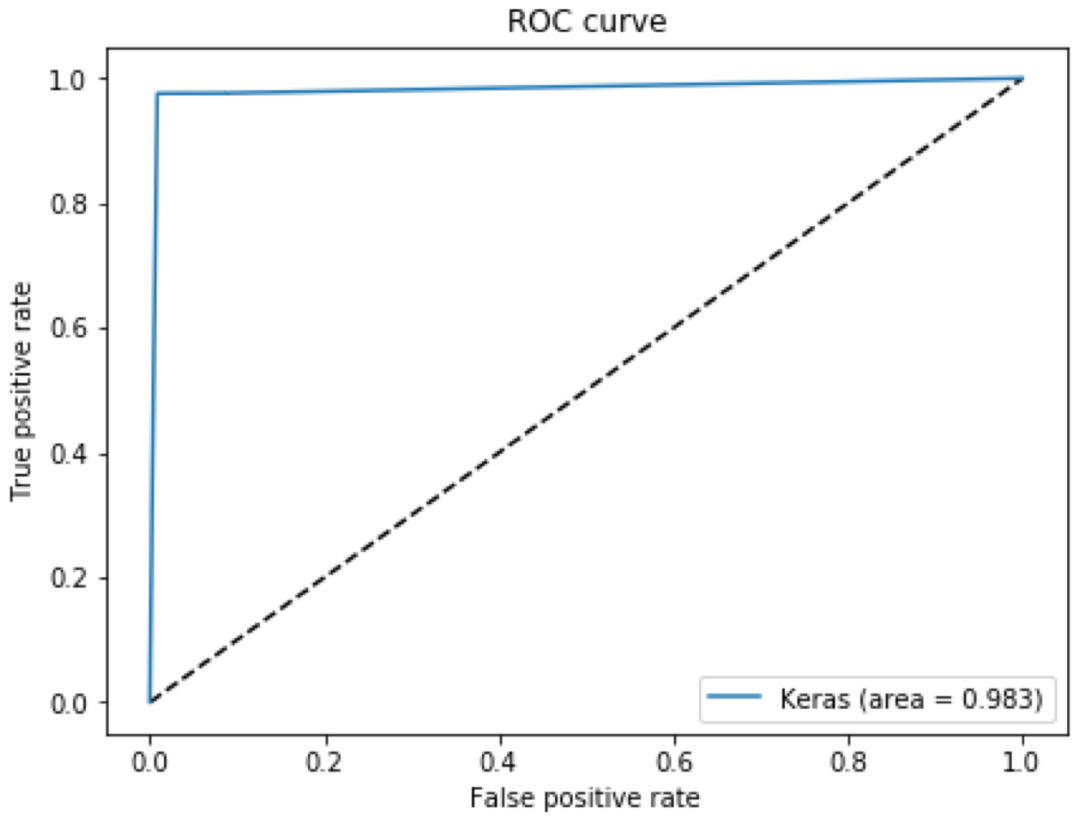
	precision	recall	f1-score	support
Normal	1.00	0.99	1.00	191126
Malicious	0.85	0.98	0.91	9573
accuracy			0.99	200699

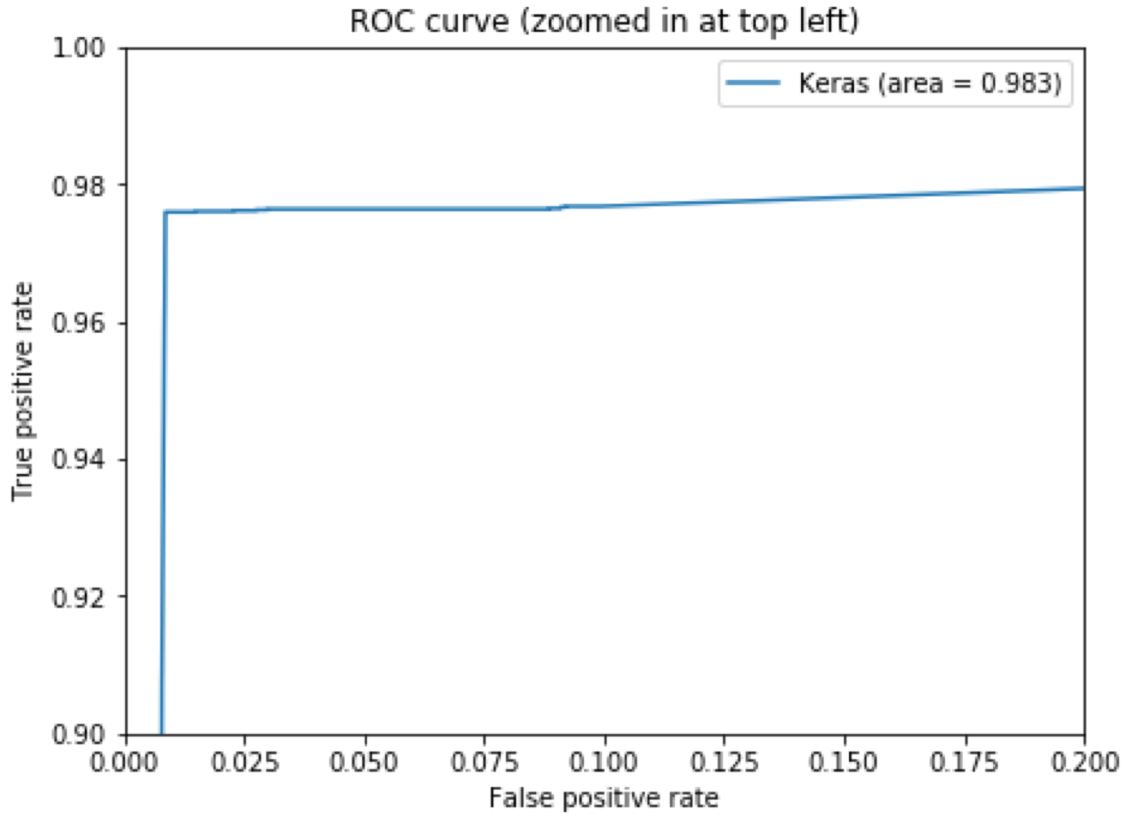




Result
Model with Modified Header

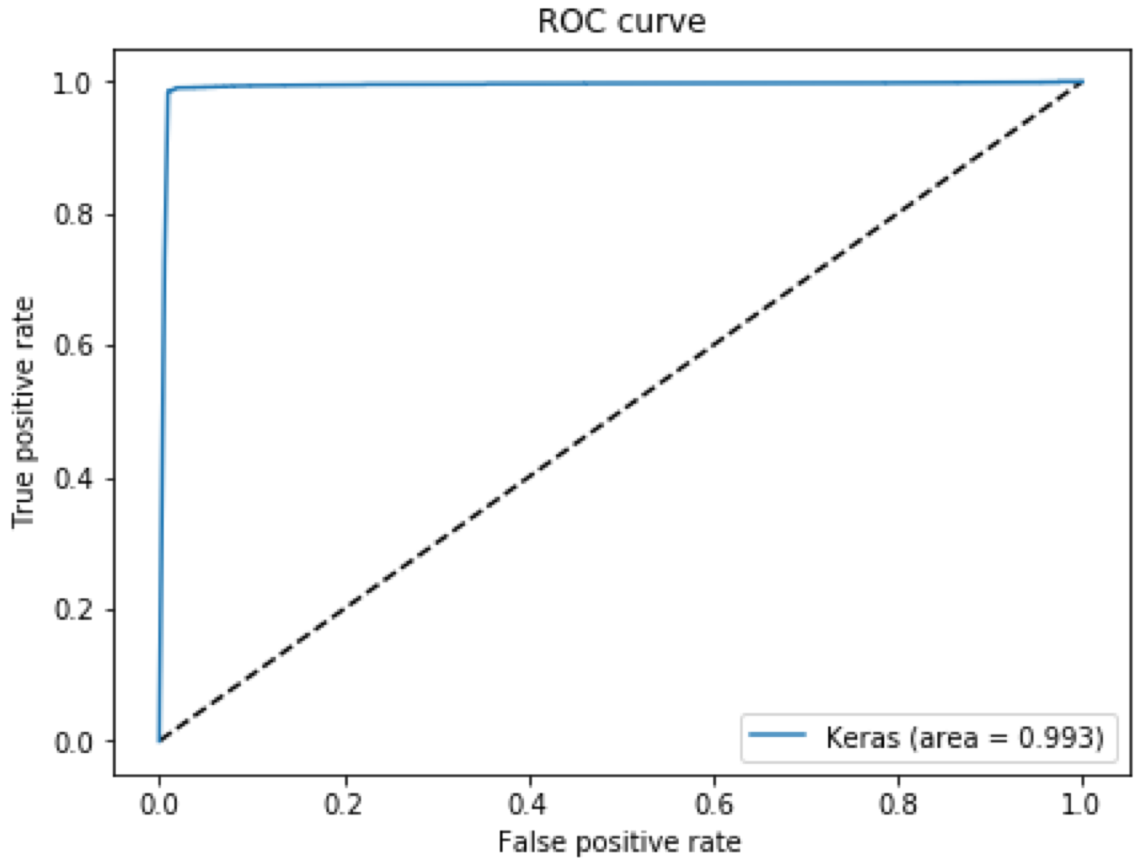
	precision	recall	f1-score	support
Normal	1.00	0.99	1.00	191126
Malicious	0.85	0.98	0.91	9573
accuracy			0.99	200699

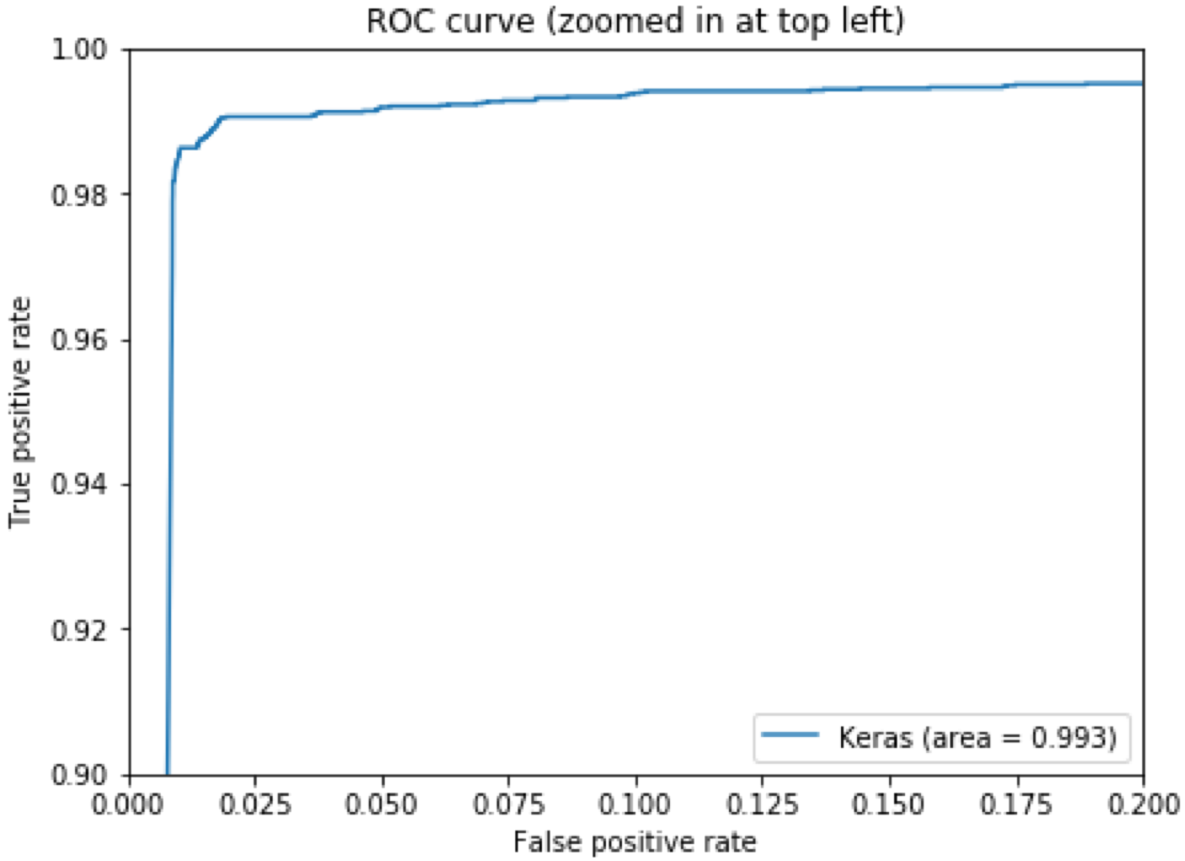




Result
Light Weight Model

	precision	recall	f1-score	support
Normal	1.00	0.99	1.00	152889
Malicious	0.85	0.98	0.91	7670
accuracy			0.99	160559

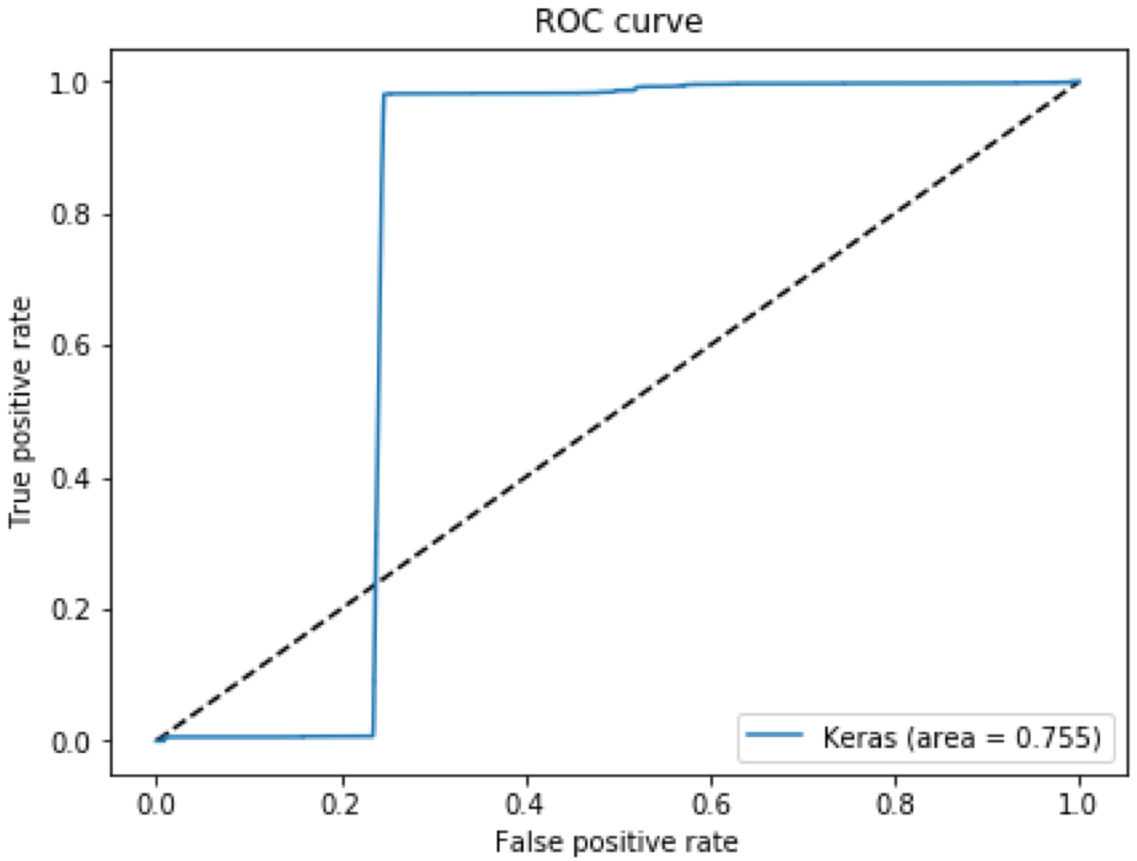


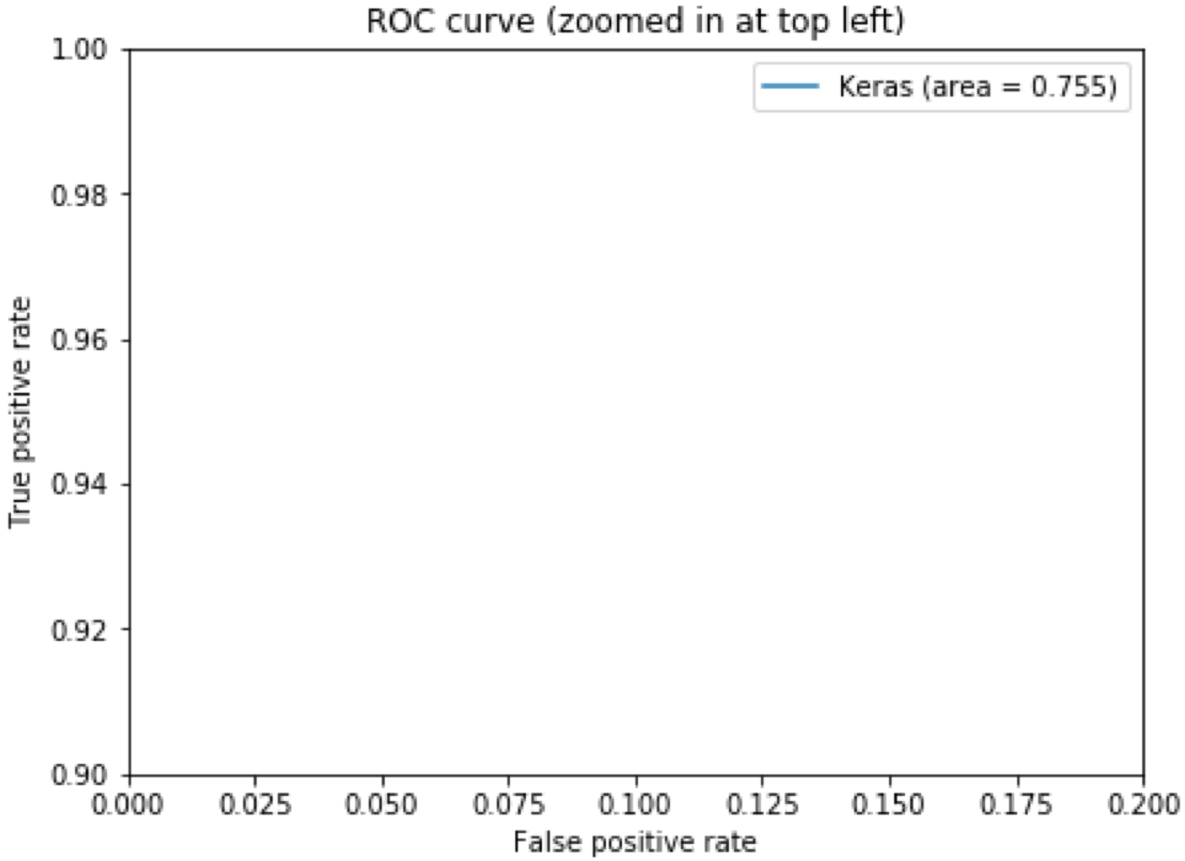


Result
Augmentation

Failed. The model predicts all samples as Normal.

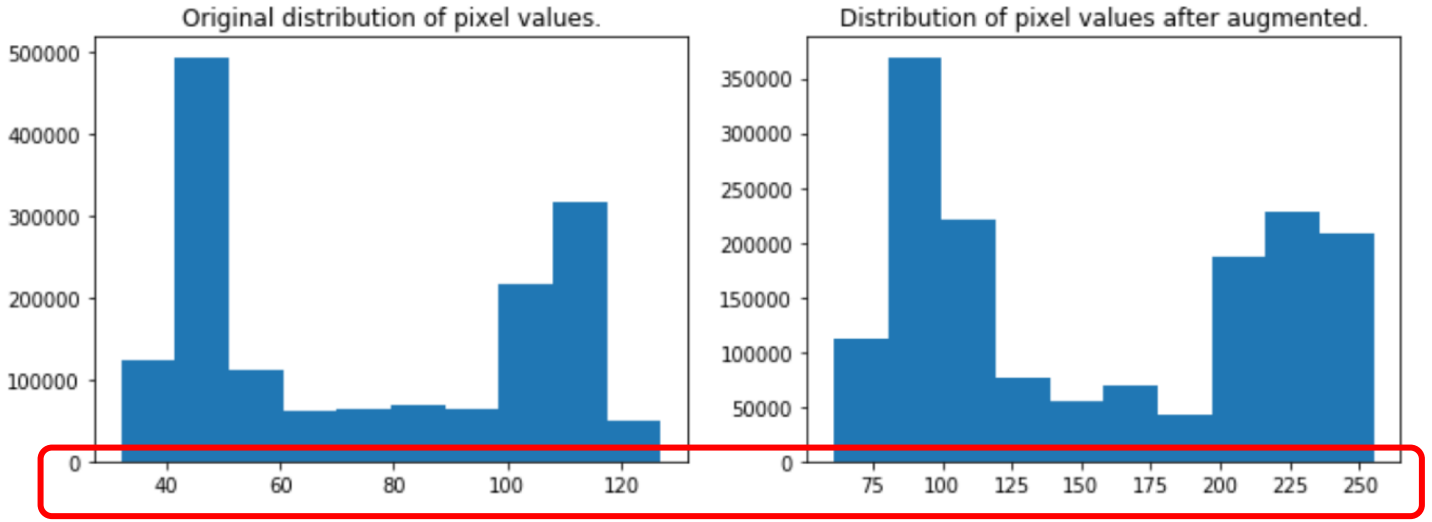
tn(True Negative) 152953, fp(False Positive) 0
fn(False Negative) 7606, tp(True Positive) 0



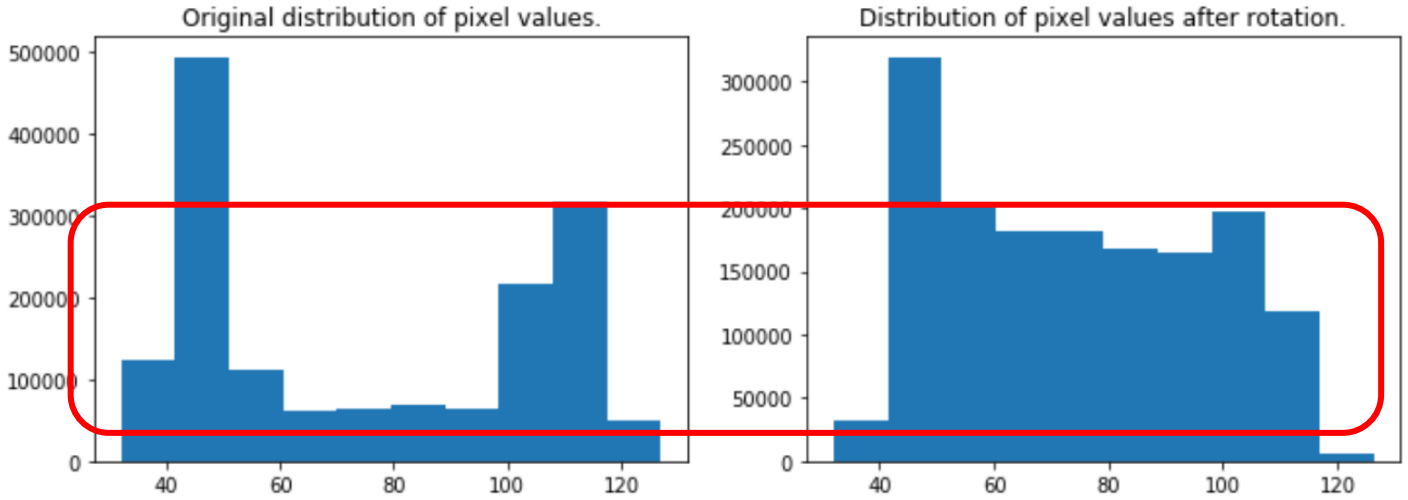


Result Augmentation

- For random value shift, the **range** of data values **expands**.
- For rotation, the data values change from **integers** into **floats**.
- All these processing methods make training data **inconsistent** with validation data and lead to such poor result.



Histograms of the original data and the data after value shift.



Histograms of the original data and the data after rotation.

Conclusion

All the data are split randomly into 80% training set and 20% validation set.

	Accuracy	AUC	Time Cost (s/epoch)
Prototype Model	**0.99066**	0.982	167
Model with Modified Header	0.99064	0.983	176
Light Weight Model	0.99056	**0.993**	**67**

- Two introduced header: **high accuracy** but long training time.
- Light Weight Model: high accuracy and relative **short** training time. Better AUC (0.993 v.s. 0.982).
- As for data augmentation
  - Regular methods: not fit this situation.
  - Need specific processing methods for ID.

Q&A

The End

Thanks for Watching