GDALR: An efficient model duplication attack on black-box Machine Learning models

BY

REWANTH COOL (SECURITY CONSULTANT) , NIKHIL JOSHI (SECURITY RESEARCHER)

About us

REWANTH COOL

Security Consultant

ML Enthusiast

Full stack developer

Speaker at HITB, CRESTCON, BSIDES

NIKHIL JOSHI

Security Researcher

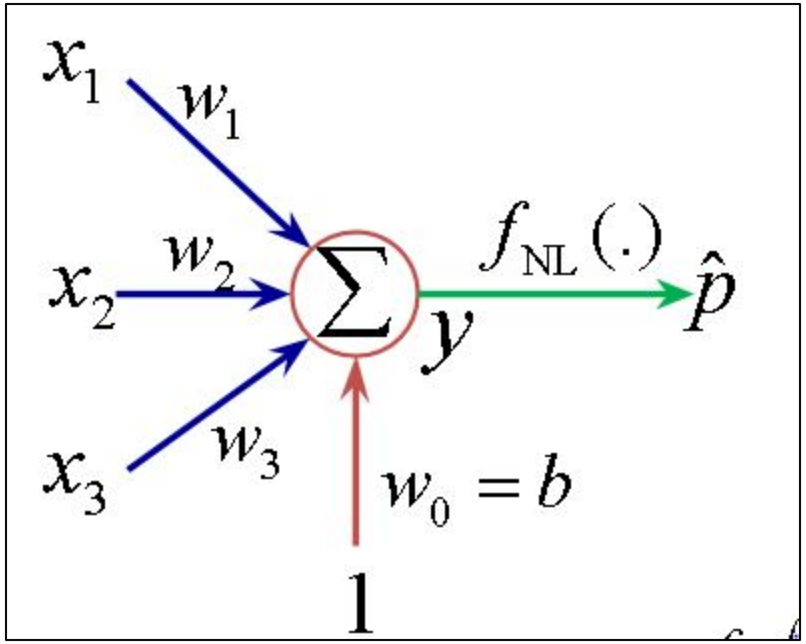
Build and Break Deep Learning systems

- About Payatu
  - A boutique security testing company specializing in IoT, Mobile, Cloud – https://payatu.com
  - In-house Fuzz testing Infrastructure
  - Mobile/Windows kernel/IoT exploitation training – Blackhat, Brucon, Hack In Paris, HITB and Corporate trainings

Agenda

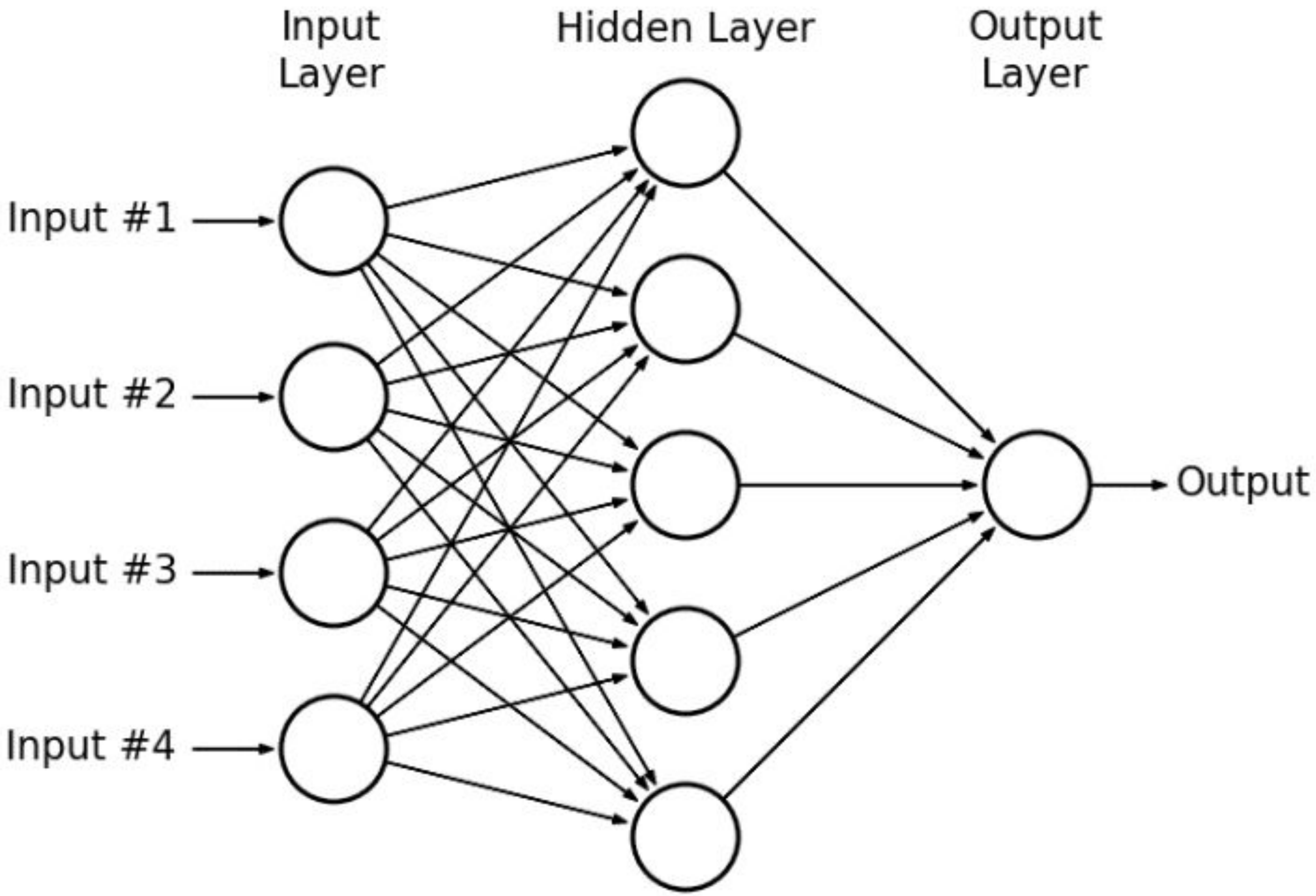
- End-to-end Machine Learning pipeline
- Model stealing/duplication techniques
- Abusing APIs to steal models deployed on cloud
- Present attack methodology
- Inefficiencies with present attack methodology
- Scope for Attack optimization
- Proposed approach (GDALR)
- Results and conclusion

PERCEPTRON



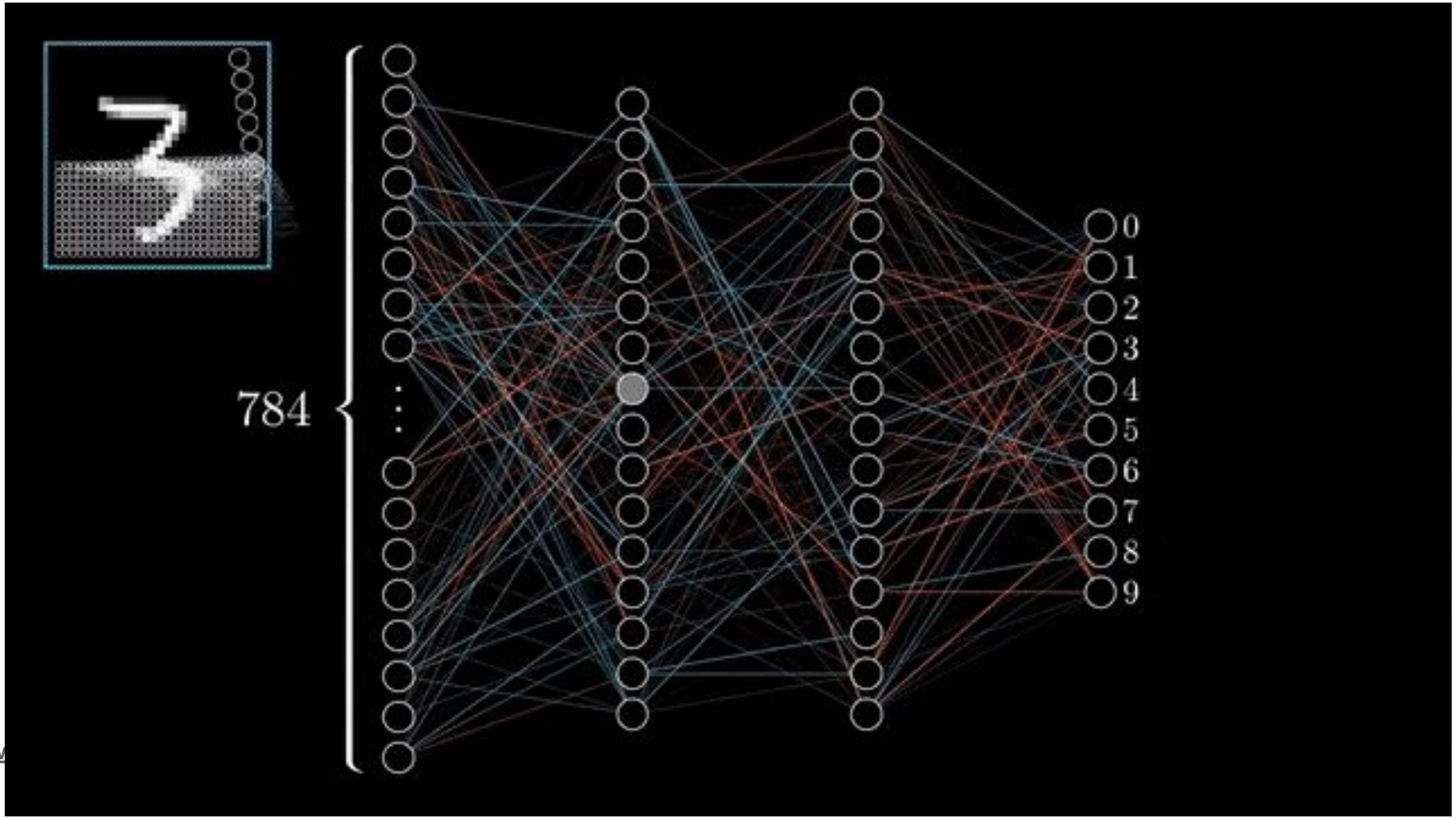
Ref: http://www.xpertup.com/2018/05/11/loss-functions-and-optimization-algorithms/

MULTI LAYER PERCEPTRON (MLP)



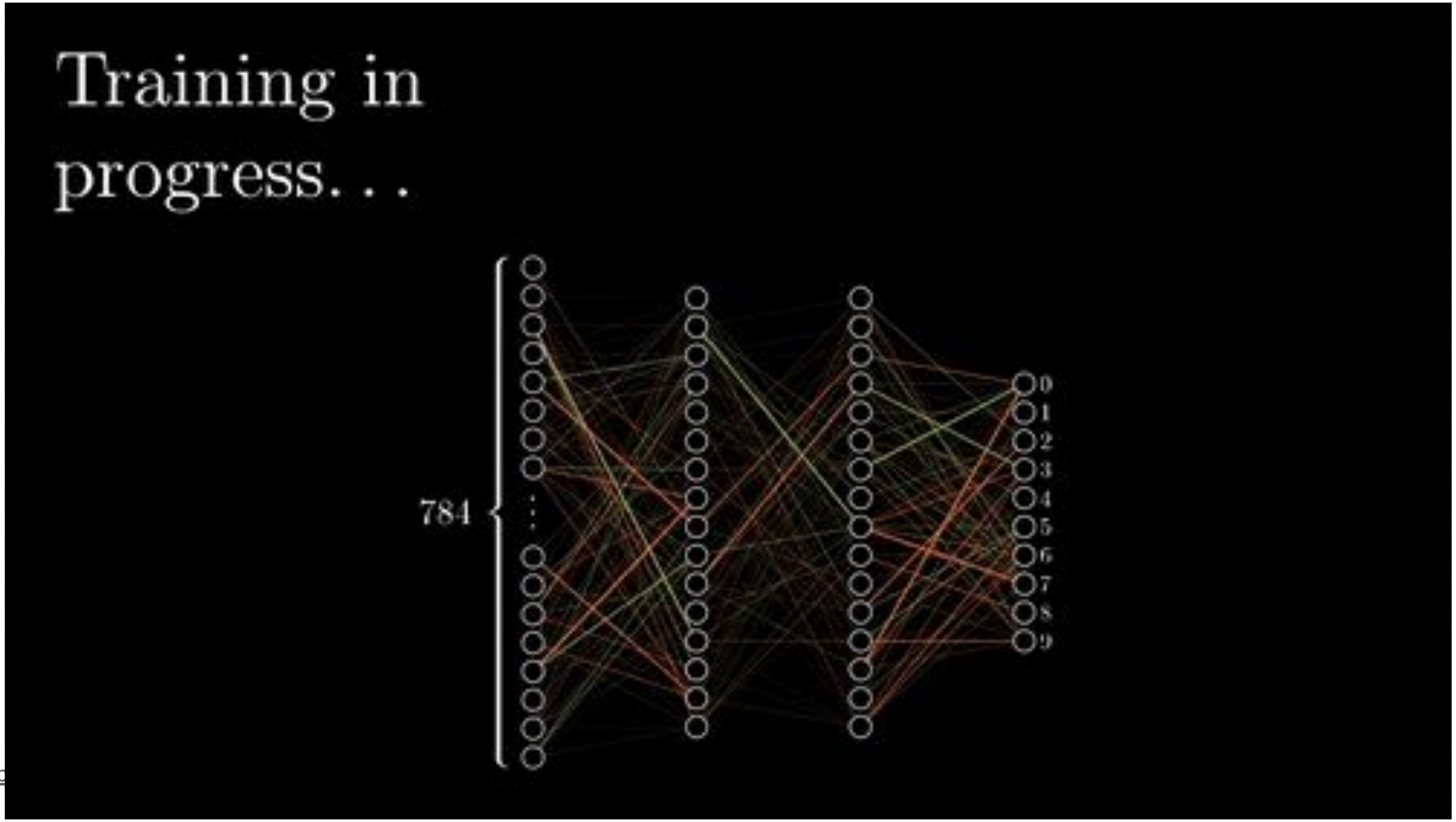
Ref: https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

MLP



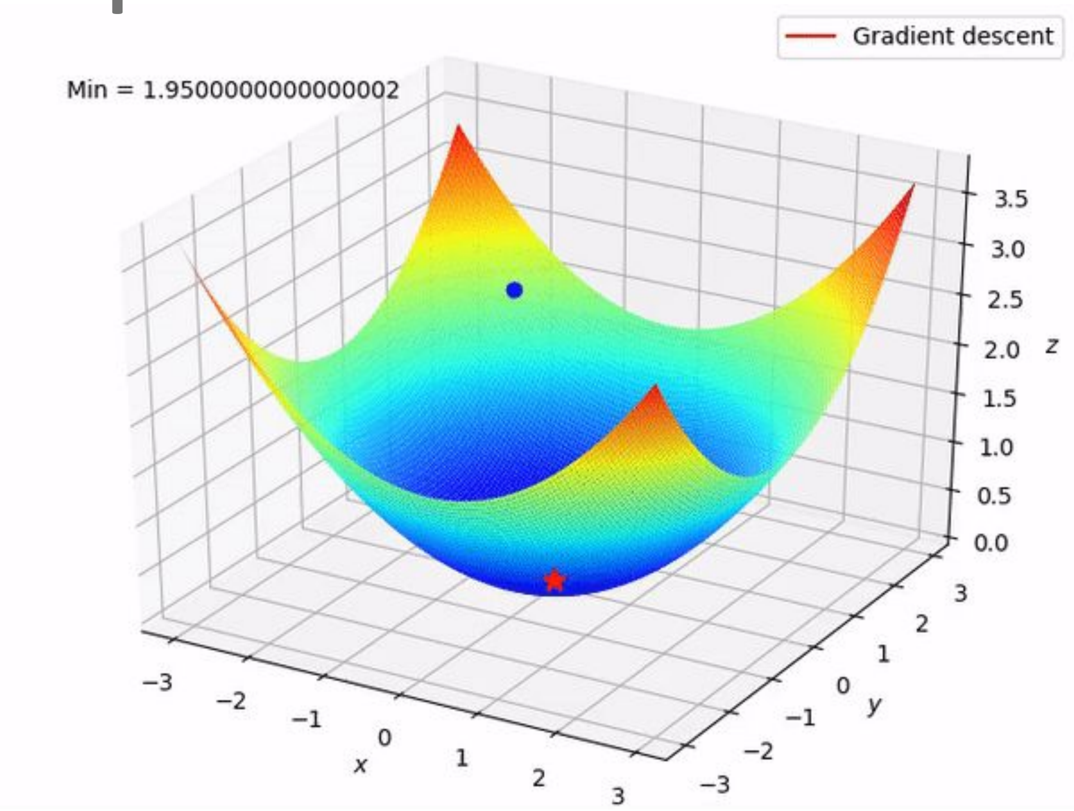
Ref: https://ww

MLP



Ref: http

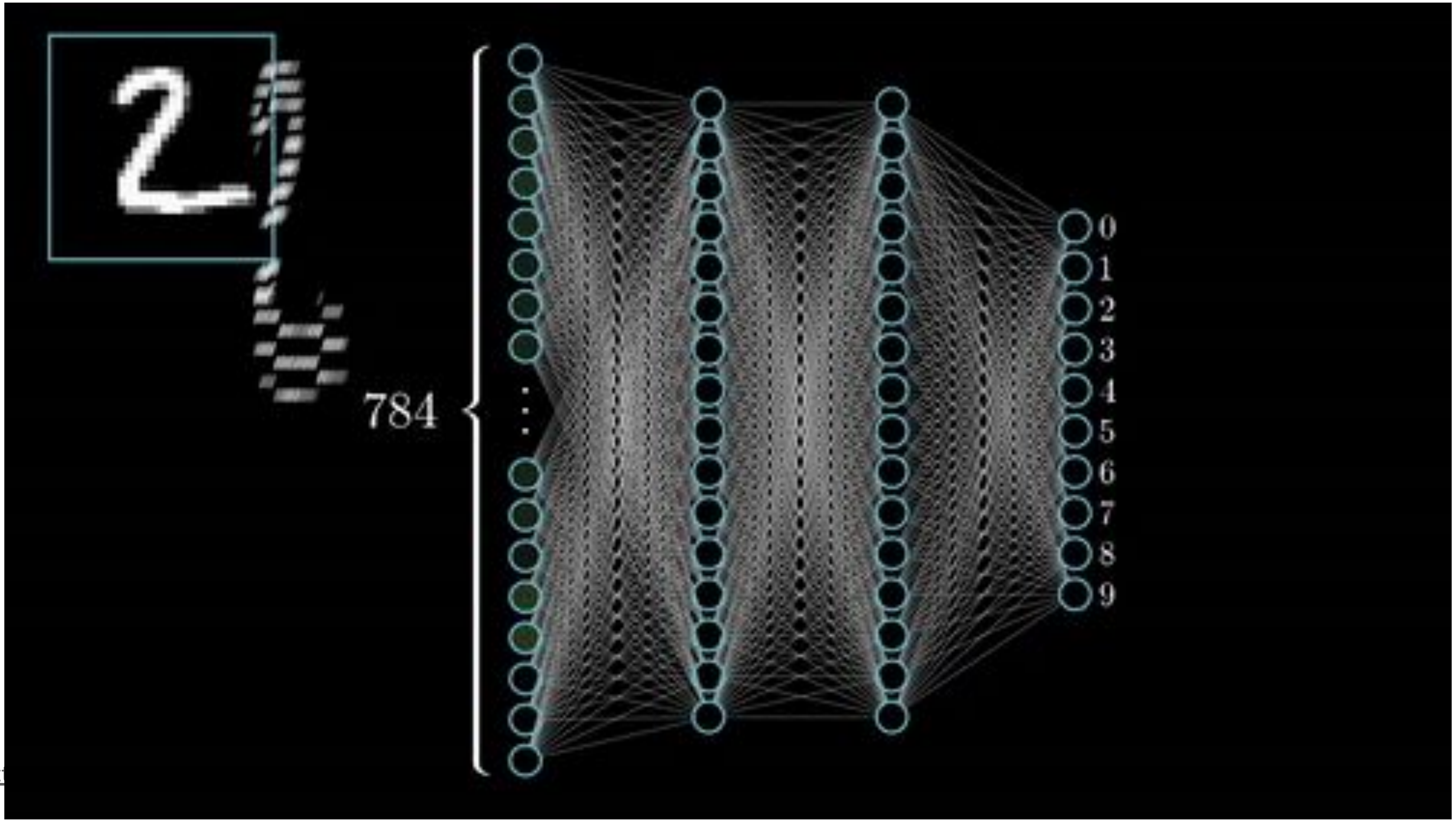
Optimization



Ref: http://www.xpertup.com/2018/05/11/loss-functions-and-optimization-algorithms/

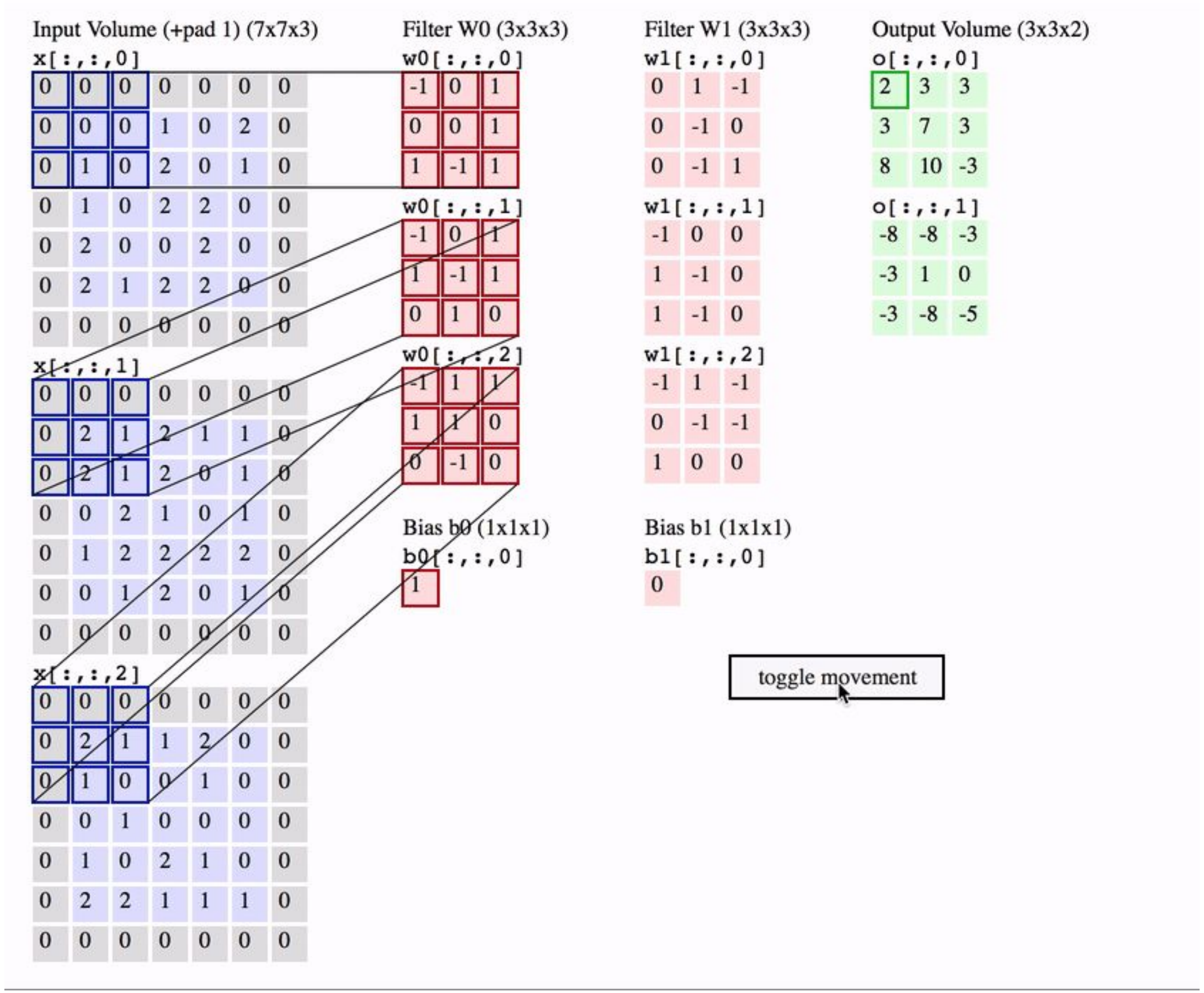
$$W_{i+1} = W_i + \alpha \frac{\partial}{\partial w} \mathbf{C}$$

MLP



Ref: ht

Convolutional Neural Networks



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

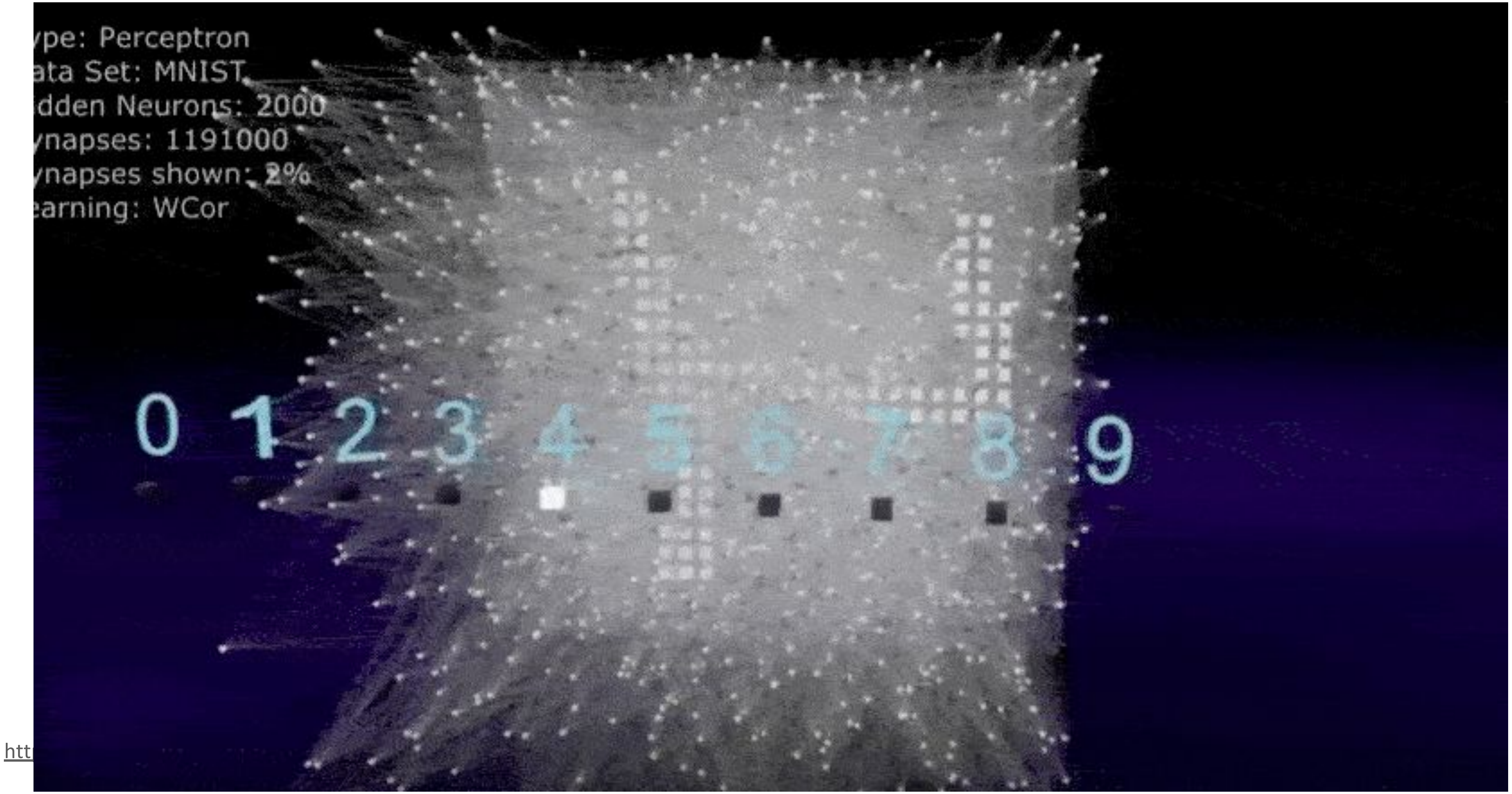
2	3	3
3	7	3
8	10	-3

o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

Ref: http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks



Ref: htt

Model stealing/duplication techniques

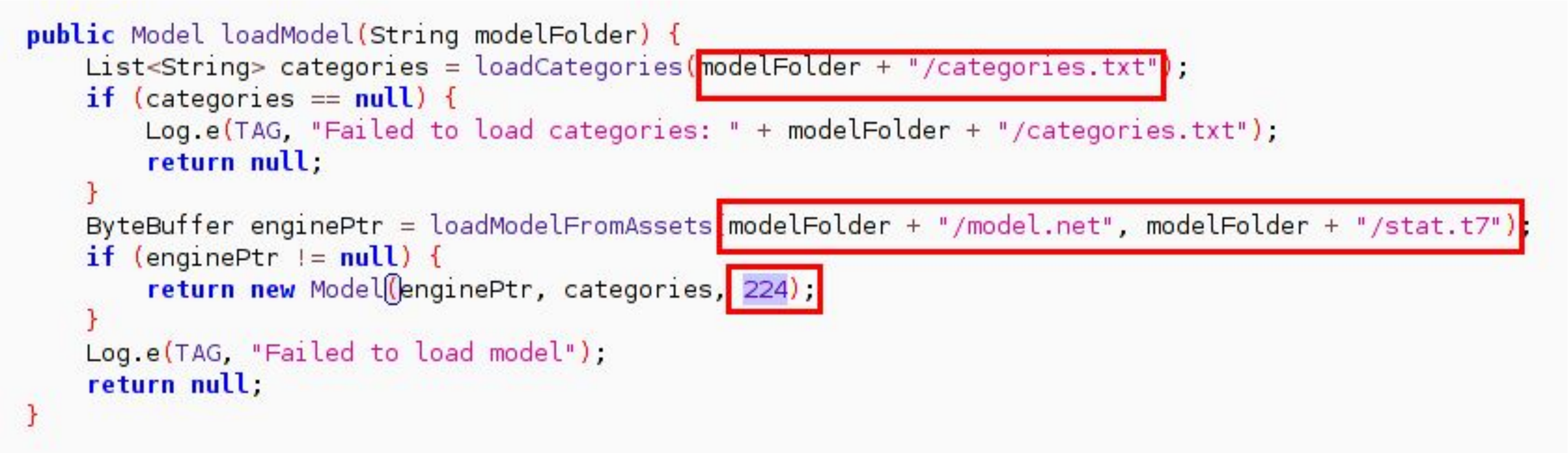
- Offline attacks
- Online attacks

Offline attacks

Steps -

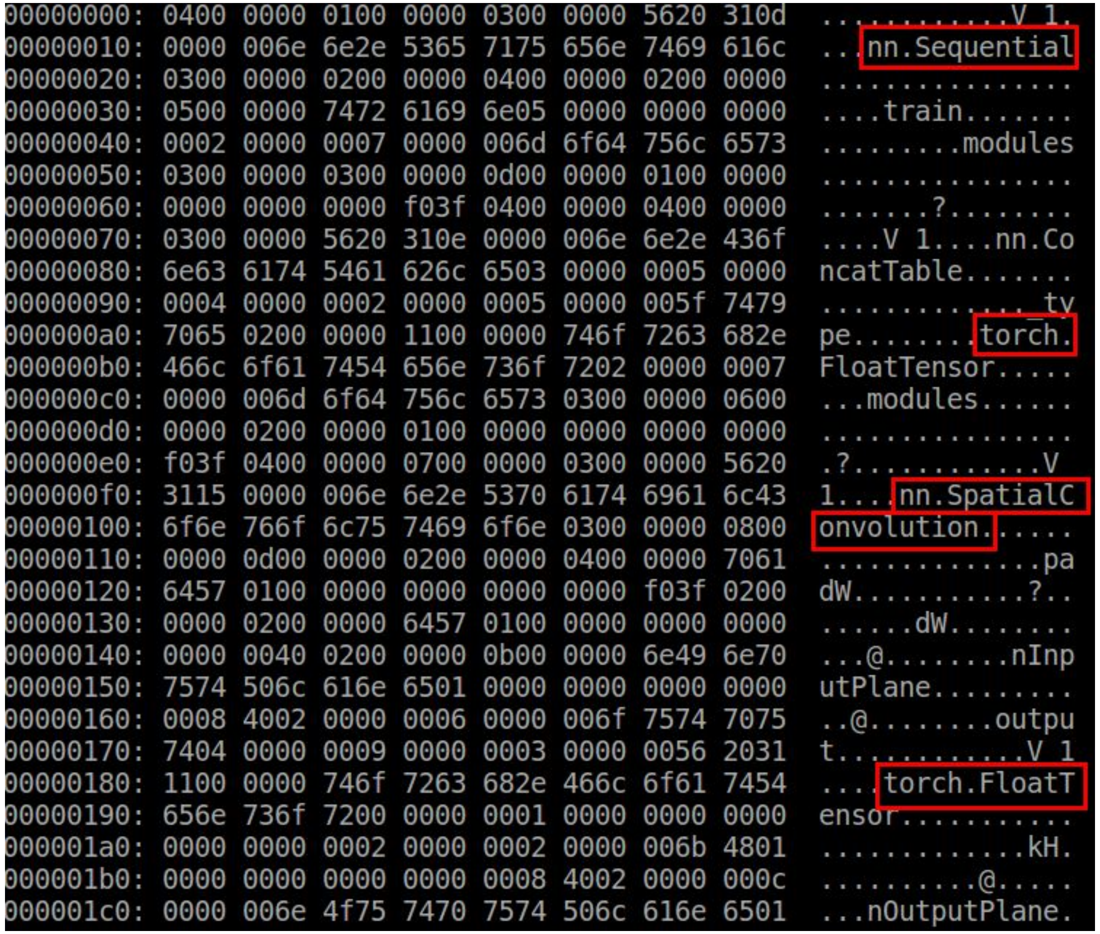
1. Reverse engineer the executable to find hidden gems
2. Locate the trained model stored on device
3. Analyse the serialized model
4. Own the model

Offline attacks



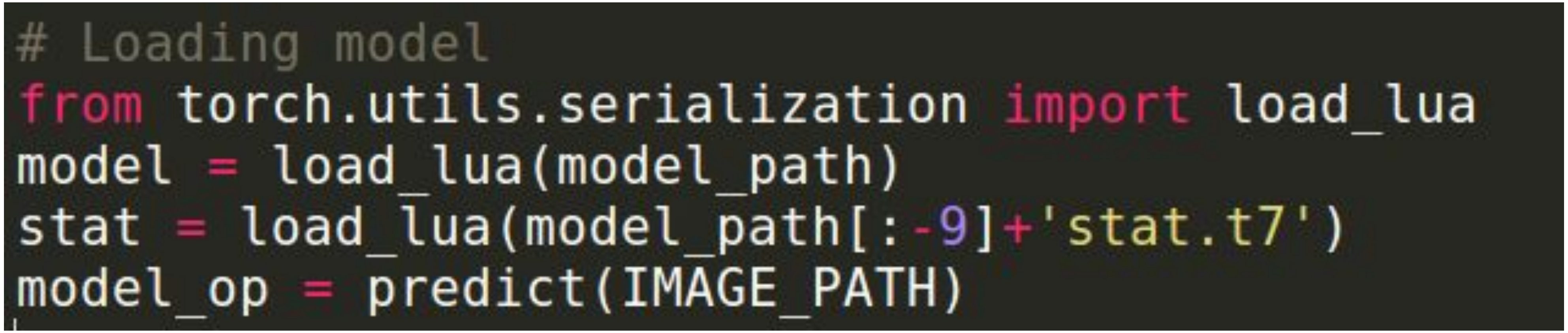
```
public Model loadModel(String modelFolder) {
    List<String> categories = loadCategories(modelFolder + "/categories.txt");
    if (categories == null) {
        Log.e(TAG, "Failed to load categories: " + modelFolder + "/categories.txt");
        return null;
    }
    ByteBuffer enginePtr = loadModelFromAssets(modelFolder + "/model.net", modelFolder + "/stat.t7");
    if (enginePtr != null) {
        return new Model(enginePtr, categories, 224);
    }
    Log.e(TAG, "Failed to load model");
    return null;
}
```

Offline attacks



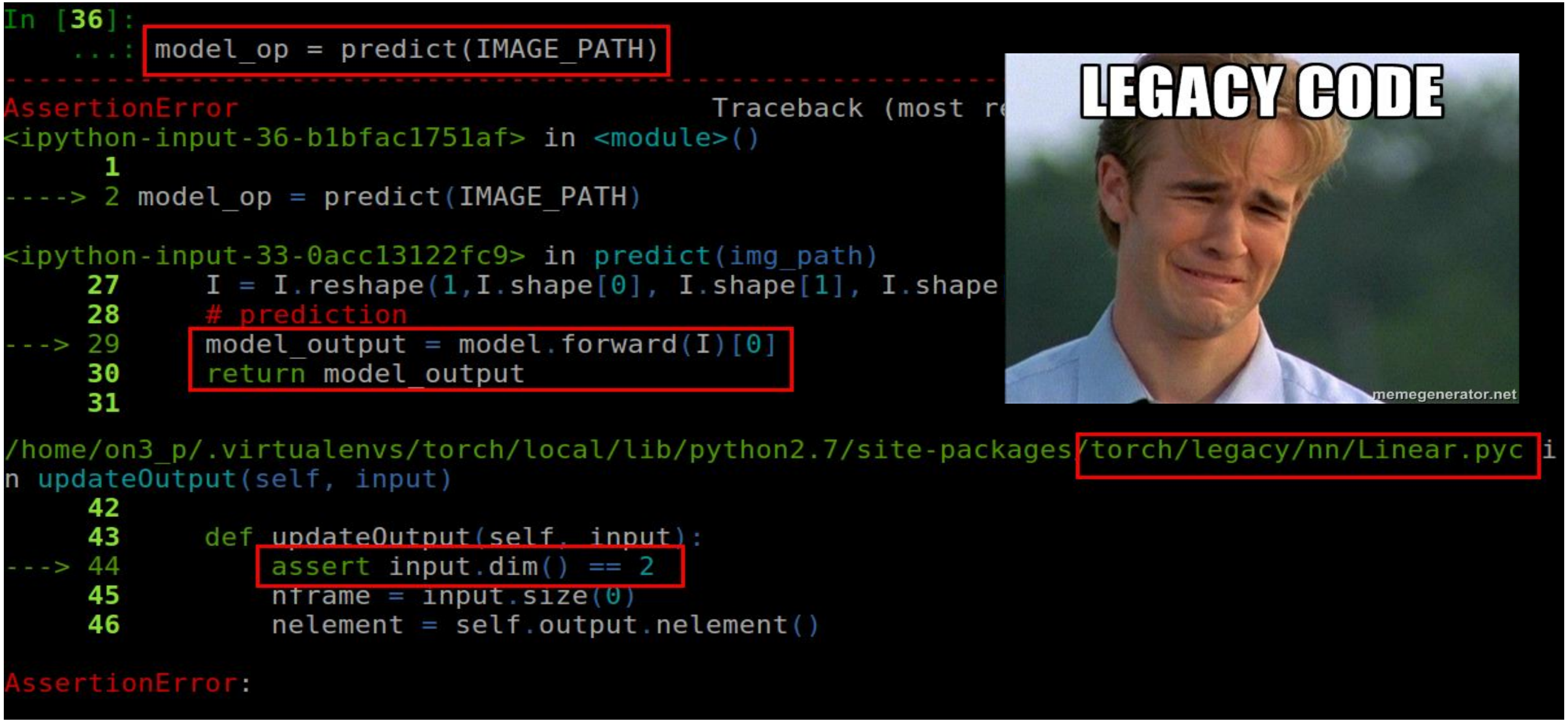
```
00000000: 0400 0000 0100 0000 0300 0000 5620 310d  ............V 1.
00000010: 0000 006e 6e2e 5365 7175 656e 7469 616c  ...nn.Sequential
00000020: 0300 0000 0200 0000 0400 0000 0200 0000  ................
00000030: 0500 0000 7472 6169 6e05 0000 0000 0000  ....train.......
00000040: 0002 0000 0007 0000 006d 6f64 756c 6573  .........modules
00000050: 0300 0000 0300 0000 0d00 0000 0100 0000  ................
00000060: 0000 0000 0000 f03f 0400 0000 0400 0000  .......?........
00000070: 0300 0000 5620 310e 0000 006e 6e2e 436f  ....V 1....nn.Co
00000080: 6e63 6174 5461 626c 6503 0000 0005 0000  ncatTable.......
00000090: 0004 0000 0002 0000 0005 0000 005f 7479  ............._ty
000000a0: 7065 0200 0000 1100 0000 746f 7263 682e  pe........torch.
000000b0: 466c 6f61 7454 656e 736f 7202 0000 0007  FloatTensor.....
000000c0: 0000 006d 6f64 756c 6573 0300 0000 0600  ...modules......
000000d0: 0000 0200 0000 0100 0000 0000 0000 0000  ................
000000e0: f03f 0400 0000 0700 0000 0300 0000 5620  .?............V 
000000f0: 3115 0000 006e 6e2e 5370 6174 6961 6c43  1....nn.SpatialC
00000100: 6f6e 766f 6c75 7469 6f6e 0300 0000 0800  onvolution......
00000110: 0000 0d00 0000 0200 0000 0400 0000 7061  ..............pa
00000120: 6457 0100 0000 0000 0000 0000 f03f 0200  dW...........?..
00000130: 0000 0200 0000 6457 0100 0000 0000 0000  ......dW........
00000140: 0000 0040 0200 0000 0b00 0000 6e49 6e70  ...@........nInp
00000150: 7574 506c 616e 6501 0000 0000 0000 0000  utPlane.........
00000160: 0008 4002 0000 0006 0000 006f 7574 7075  ..@........outpu
00000170: 7404 0000 0009 0000 0003 0000 0056 2031  t............V 1
00000180: 1100 0000 746f 7263 682e 466c 6f61 7454  ....torch.FloatT
00000190: 656e 736f 7200 0000 0001 0000 0000 0000  ensor...........
000001a0: 0000 0000 0002 0000 0002 0000 006b 4801  .............kH.
000001b0: 0000 0000 0000 0000 0008 4002 0000 000c  ..........@.....
000001c0: 0000 006e 4f75 7470 7574 506c 616e 6501  ...nOutputPlane.
```

Offline attacks



```
# Loading model
from torch.utils.serialization import load_lua
model = load_lua(model_path)
stat = load_lua(model_path[:-9]+'stat.t7')
model_op = predict(IMAGE_PATH)
```

Offline attacks

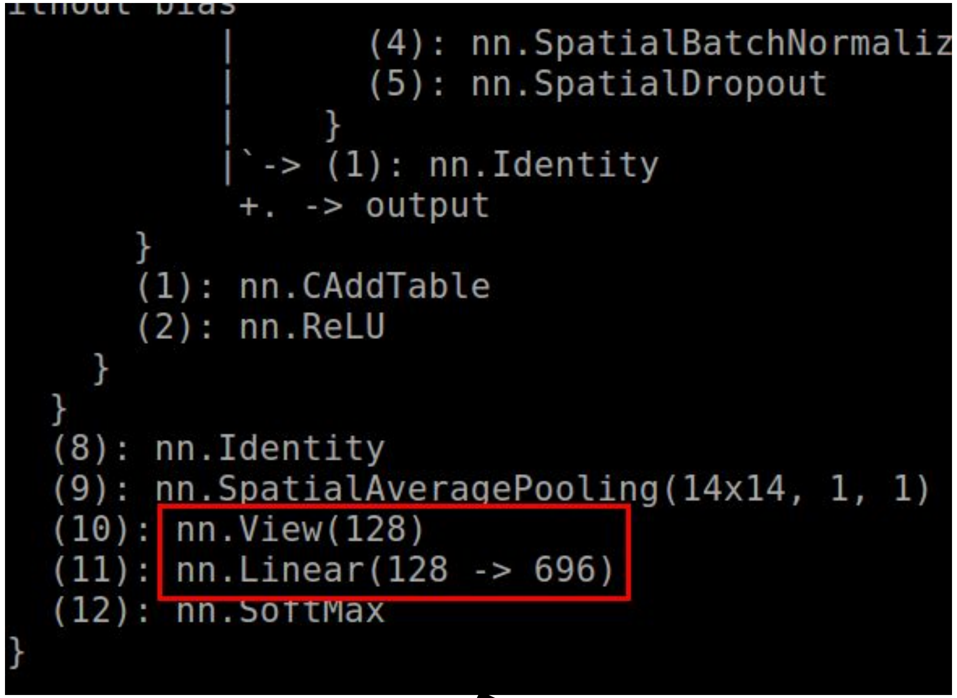


```
In [36]:
    ...: model_op = predict(IMAGE_PATH)
------------------------------------------------------------------------
AssertionError                            Traceback (most re
<ipython-input-36-b1bfac1751af> in <module>()
      1
----> 2 model_op = predict(IMAGE_PATH)

<ipython-input-33-0acc13122fc9> in predict(img_path)
     27     I = I.reshape(1,I.shape[0], I.shape[1], I.shape
     28     # prediction
---> 29     model_output = model.forward(I)[0]
     30     return model_output
     31

/home/on3_p/.virtualenvs/torch/local/lib/python2.7/site-packages/torch/legacy/nn/Linear.pyc i
n updateOutput(self, input)
     42
     43     def updateOutput(self, input):
---> 44         assert input.dim() == 2
     45         nframe = input.size(0)
     46         nelement = self.output.nelement()

AssertionError:
```

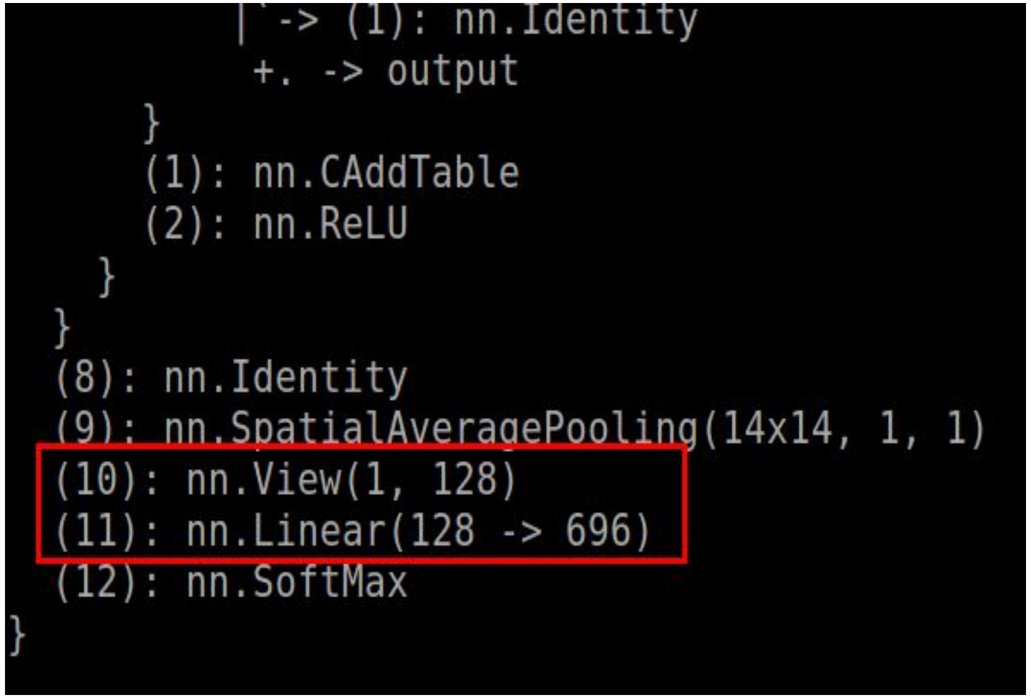
Offline attacks

DEBUGGING
I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK, BUT I WILL FIND YOU, AND
I WILL FIX YOU

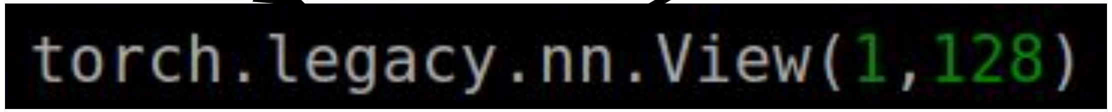
Offline attacks



```
ithout bias
                |        (4): nn.SpatialBatchNormaliz
                |        (5): nn.SpatialDropout
                |     }
                |`-> (1): nn.Identity
                 +. -> output
          }
          (1): nn.CAddTable
          (2): nn.ReLU
        }
    }
    (8): nn.Identity
    (9): nn.SpatialAveragePooling(14x14, 1, 1)
    (10): nn.View(128)
    (11): nn.Linear(128 -> 696)
    (12): nn.SoftMax
}
```

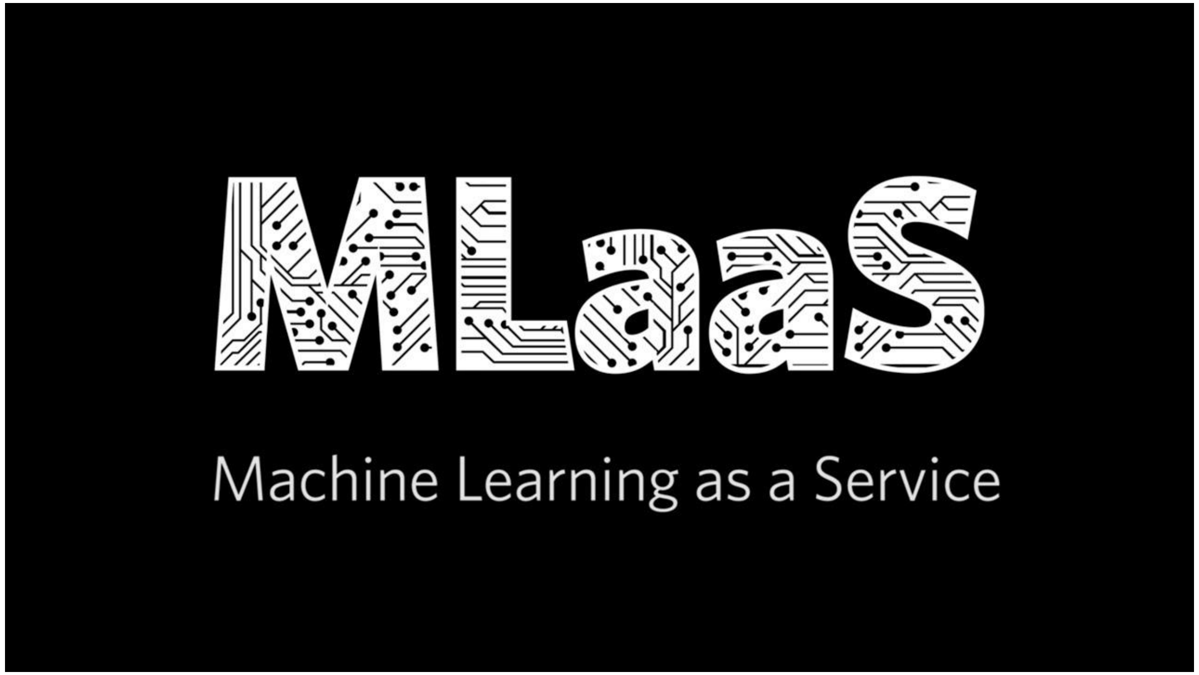


```
            |`-> (1): nn.Identity
             +. -> output
        }
        (1): nn.CAddTable
        (2): nn.ReLU
      }
    }
    (8): nn.Identity
    (9): nn.SpatialAveragePooling(14x14, 1, 1)
    (10): nn.View(1, 128)
    (11): nn.Linear(128 -> 696)
    (12): nn.SoftMax
}
```



```
torch.legacy.nn.View(1,128)
```

Offline attacks



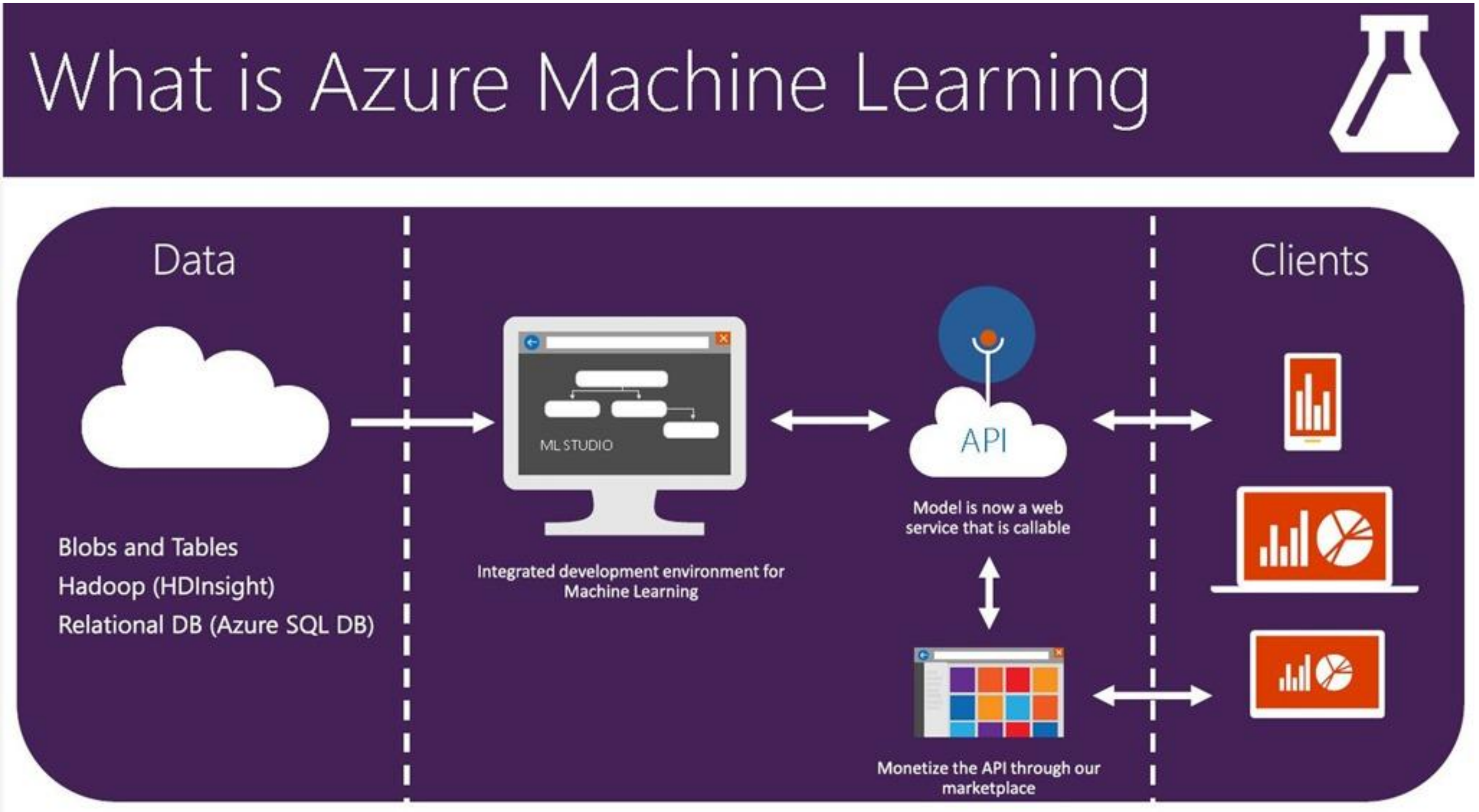
MLaaS

Machine Learning as a Service

MLaaS Service Providers

amazon
web services
Google Cloud Platform
bigml
Microsoft Azure
IBM

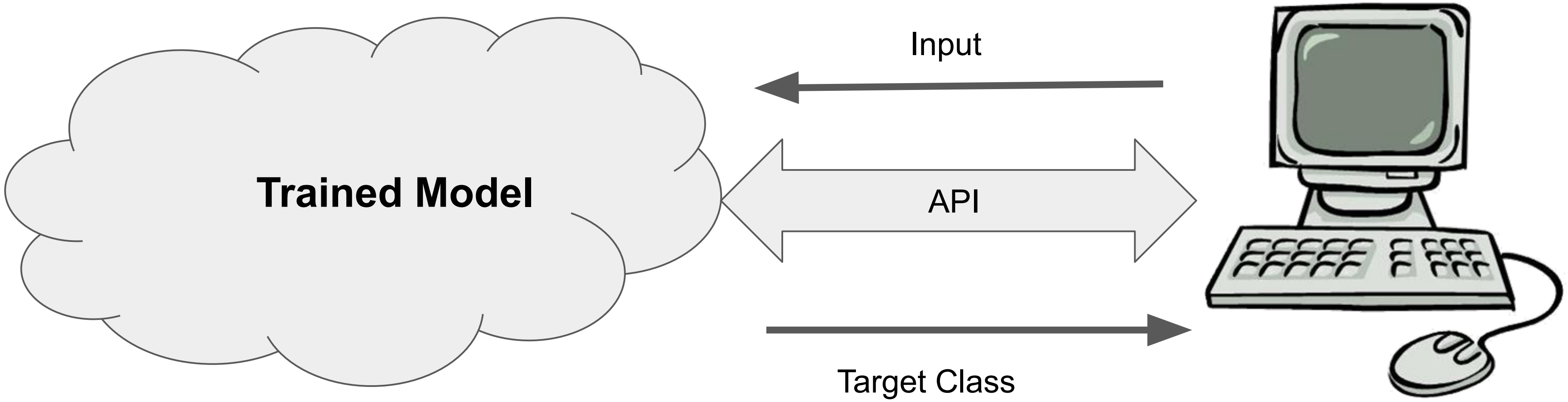
Azure ML business model



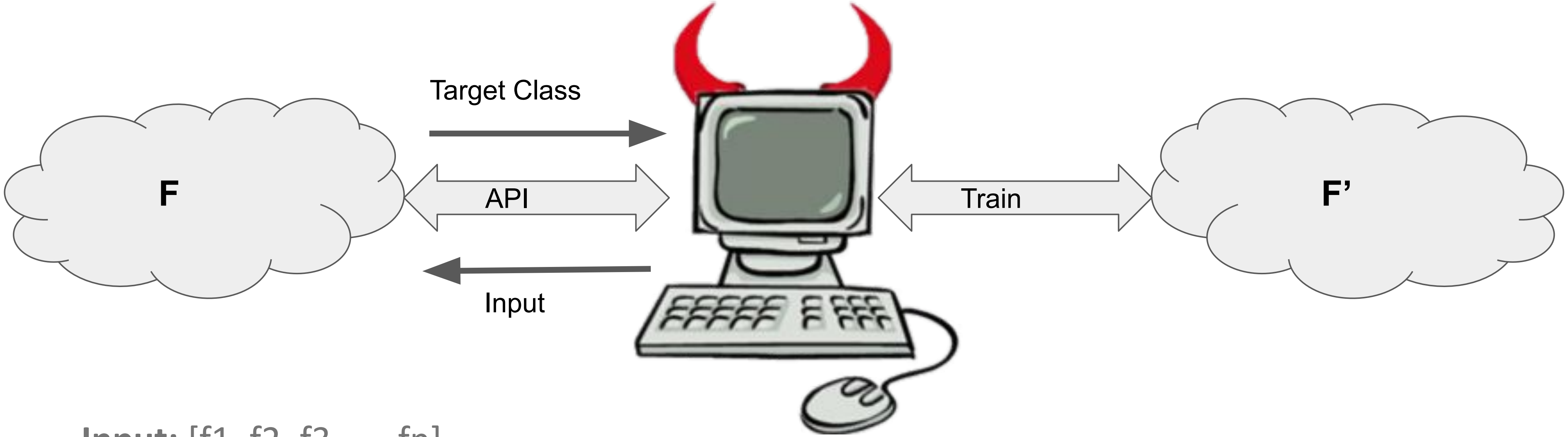
What is Azure Machine Learning
Data
Clients
ML STUDIO
API
Blobs and Tables
Hadoop (HDInsight)
Relational DB (Azure SQL DB)
Integrated development environment for Machine Learning
Model is now a web service that is callable
Monetize the API through our marketplace

Ref: **https:/**

Online attacks



Online attacks

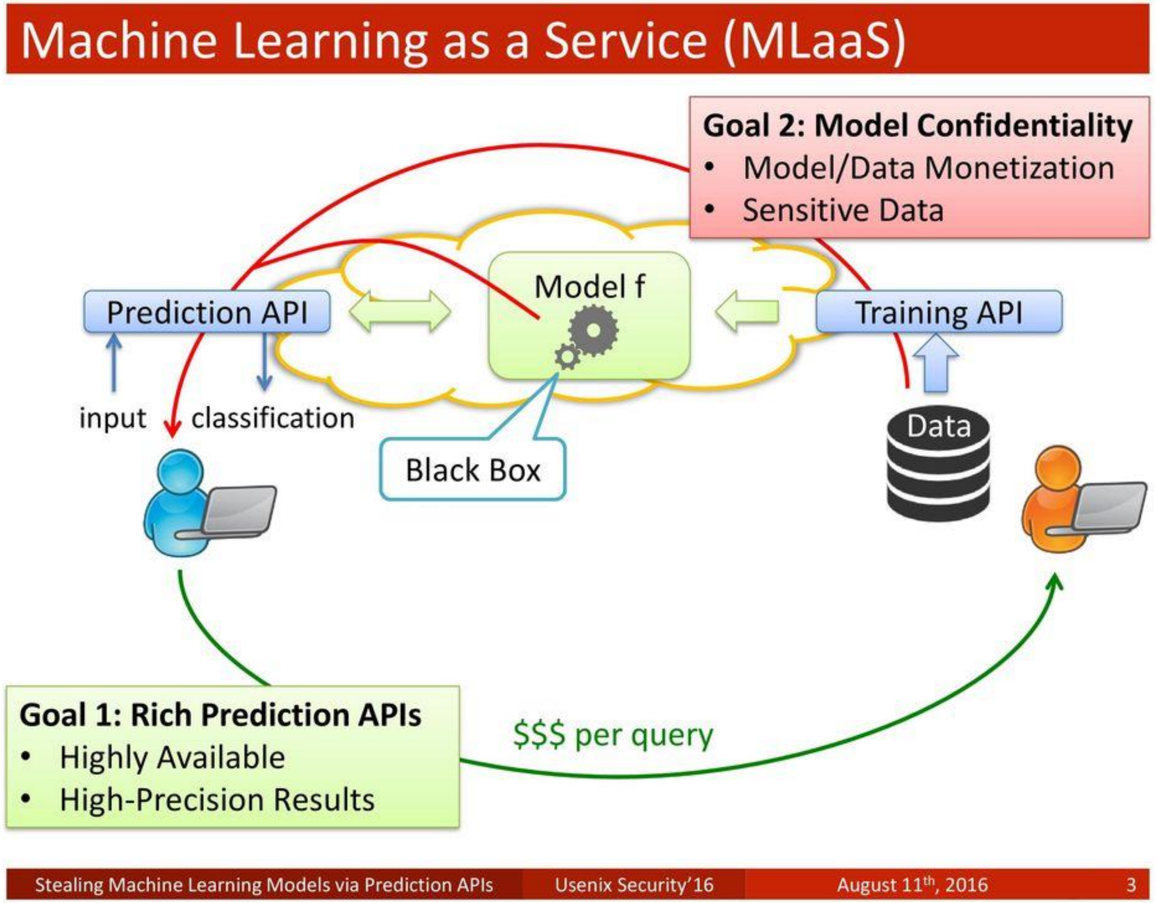


Input: [f1, f2, f3, ..., fn]

Internal F output: P(class1), P(class2), P(class3), …, P(classN)

Cloud API output: max(P(class1), P(class2), P(class3), …, P(classN))

Present attack methodology



Machine Learning as a Service (MLaaS)

Goal 2: Model Confidentiality
- Model/Data Monetization
- Sensitive Data

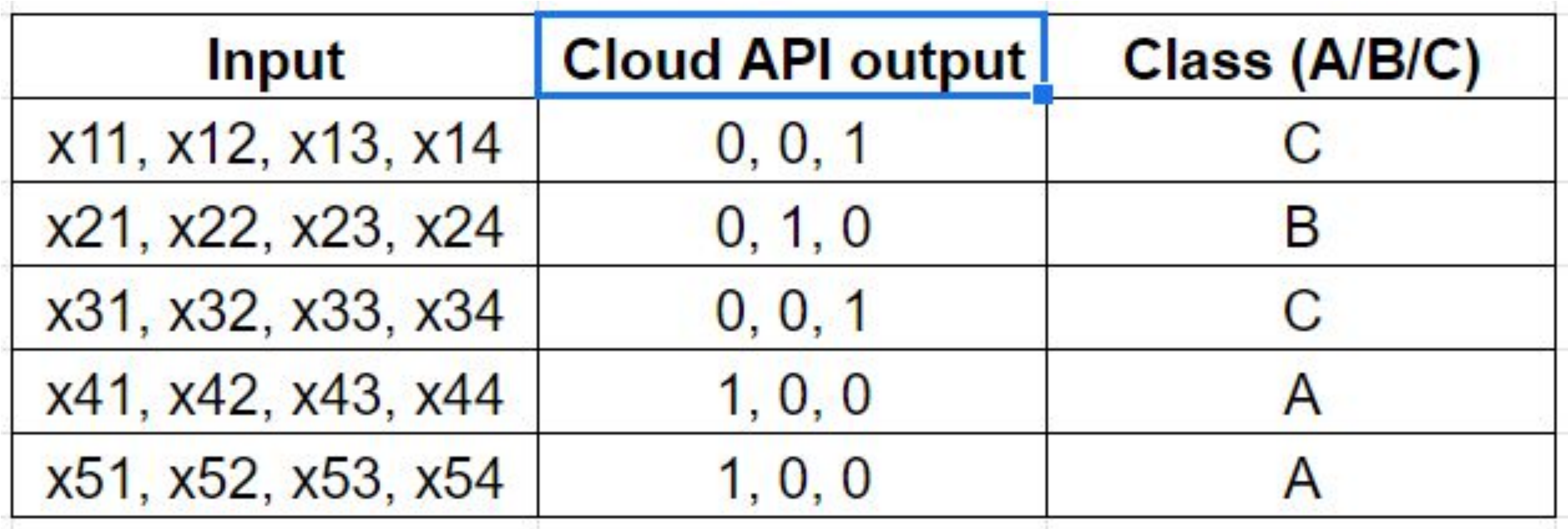
Goal 1: Rich Prediction APIs
- Highly Available
- High-Precision Results

Stealing Machine Learning Models via Prediction APIs | Usenix Security'16 | August 11th, 2016 | 3

Ref: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

Present attack methodology

In traditional approach, attackers train their local models based on Cloud API output



Input	Cloud API output	Class (A/B/C)
x11, x12, x13, x14	0, 0, 1	C
x21, x22, x23, x24	0, 1, 0	B
x31, x32, x33, x34	0, 0, 1	C
x41, x42, x43, x44	1, 0, 0	A
x51, x52, x53, x54	1, 0, 0	A

Inefficiencies with present attack methodology

Assumptions made by traditional/present attack methodology

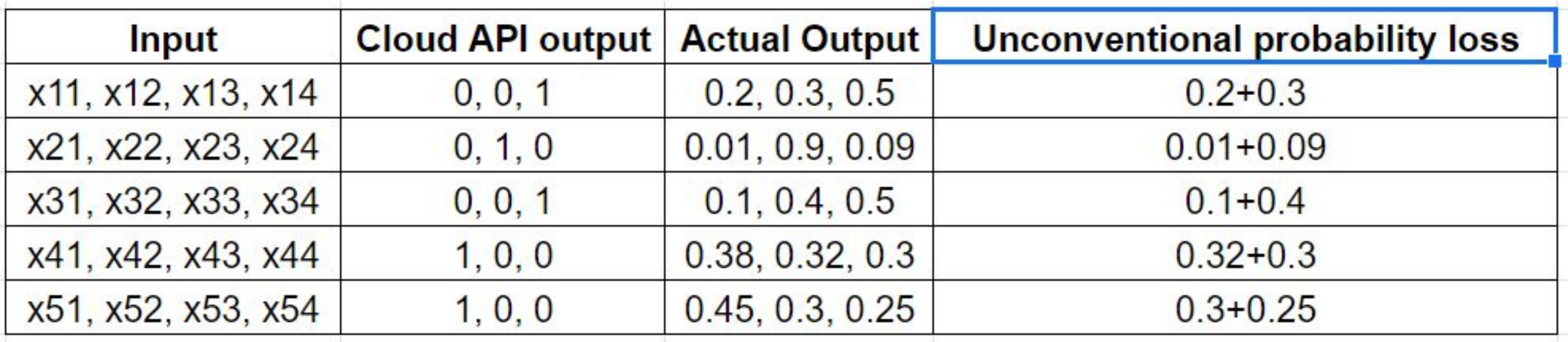
Input -
[1, 2, 3, 4]

Actual Output -
[0.3, 0.2, 0.5]

Output by Cloud API -
[0, 0, 1]

Assumption -
[0, 0, 1] $\approx$ [0.3, 0.2, 0.5]

Inefficiencies with present attack methodology



Input	Cloud API output	Actual Output	Unconventional probability loss
x11, x12, x13, x14	0, 0, 1	0.2, 0.3, 0.5	0.2+0.3
x21, x22, x23, x24	0, 1, 0	0.01, 0.9, 0.09	0.01+0.09
x31, x32, x33, x34	0, 0, 1	0.1, 0.4, 0.5	0.1+0.4
x41, x42, x43, x44	1, 0, 0	0.38, 0.32, 0.3	0.32+0.3
x51, x52, x53, x54	1, 0, 0	0.45, 0.3, 0.25	0.3+0.25

Scope for Attack optimization

1. Reconsider the way to analyze labels

Having access to all the probability values will definitely help us to clone models in an efficient way

2. Learning parameters in hyperspace

- * To Duplicate the target model we need to learn the boundaries that the target model has learnt
- * Considering probability of predicted class as 1 and others to be 0 will cause unwanted loss and increase the gradient
- * Increased gradients cause the optimizer to change weights abruptly

$$W_{i+1} = W_i + \alpha \frac{\partial}{\partial w} \mathbf{C} \uparrow$$

Proposed approach (GDALR)

GDALR: Gradient Driven Adaptive Learning Rate

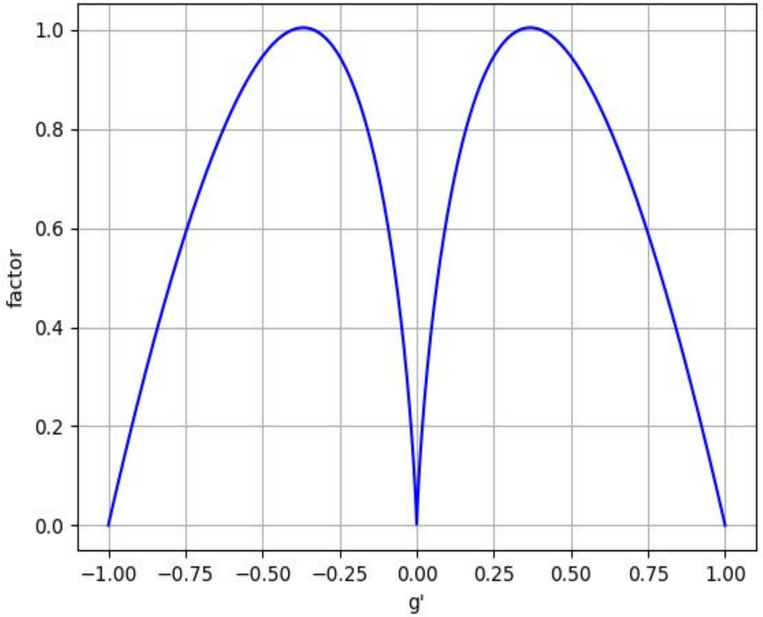
$$W_{i+1} = W_i + \updownarrow\alpha \frac{\partial}{\partial w}\mathbf{C}$$

Mathematical modification to current attack methodology

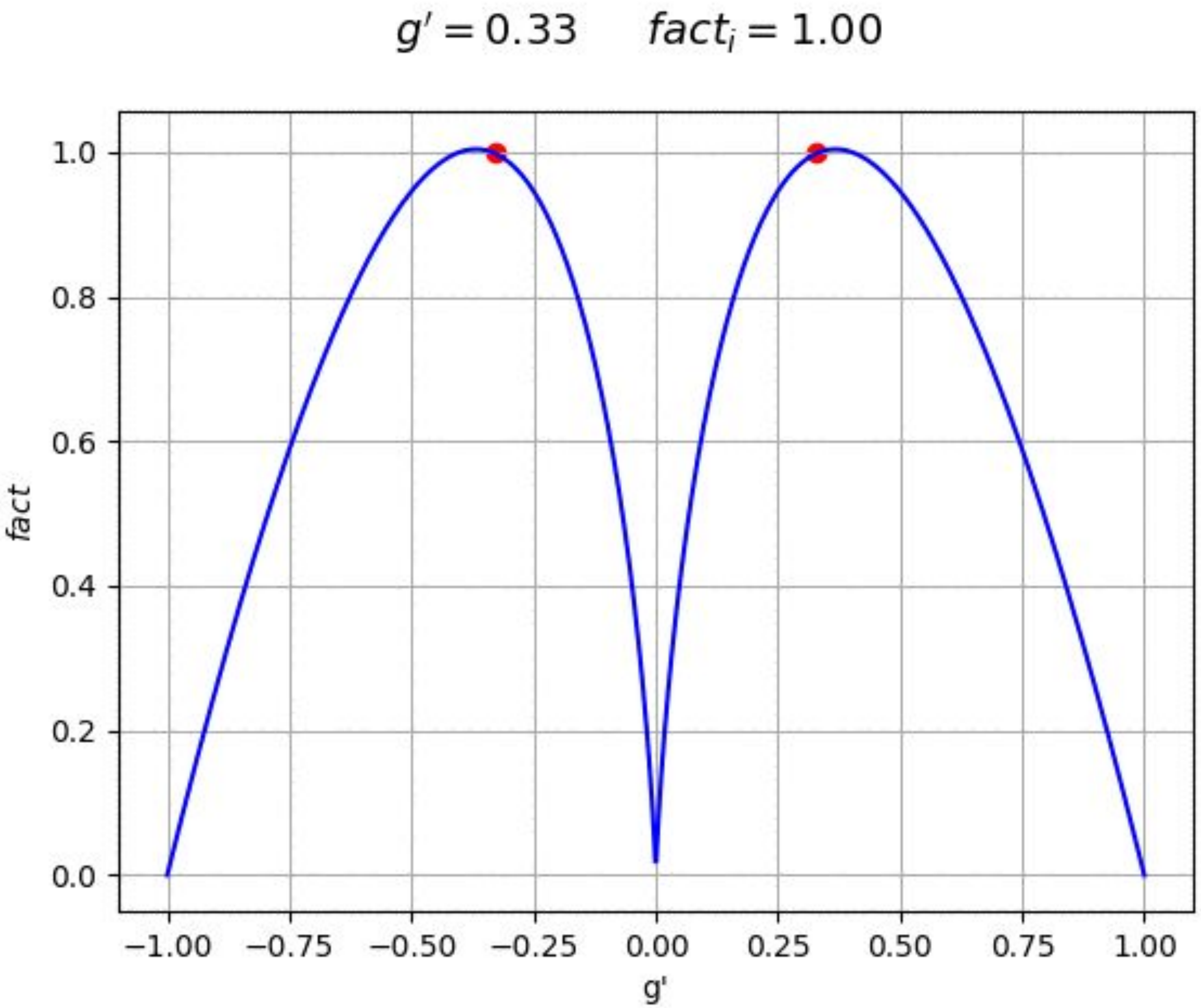
$$g_i' = \tanh(g_i) \tag{7}$$

$$fact_i = \text{abs}\left(g_i' 2\pi \log_{10}\left(\text{abs}\left(g_i'\right)\right)\right) \tag{8}$$

$$l_i' = l_i \cdot fact_i \tag{9}$$



GDALR in Action

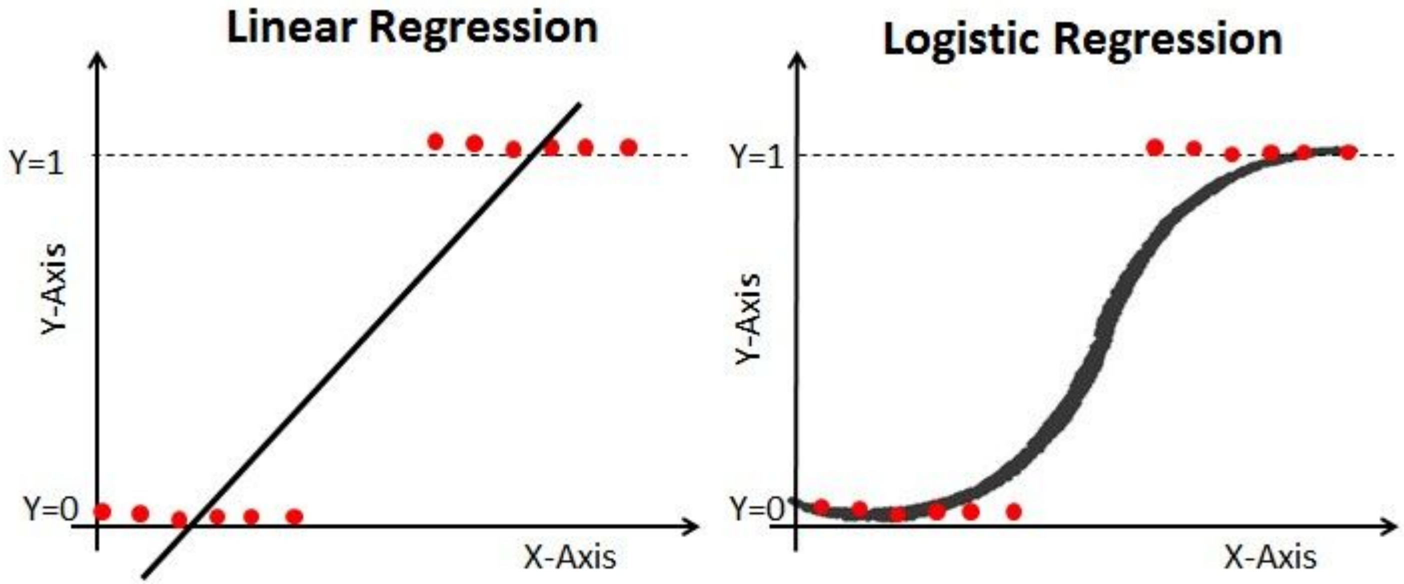


Experimental setup

GDALR has been tested on multiple classifiers -

- LOGISTIC REGRESSION
- MULTI LAYER PERCEPTRON
- CONVOLUTIONAL NEURAL NETs

LOGISTIC REGRESSION

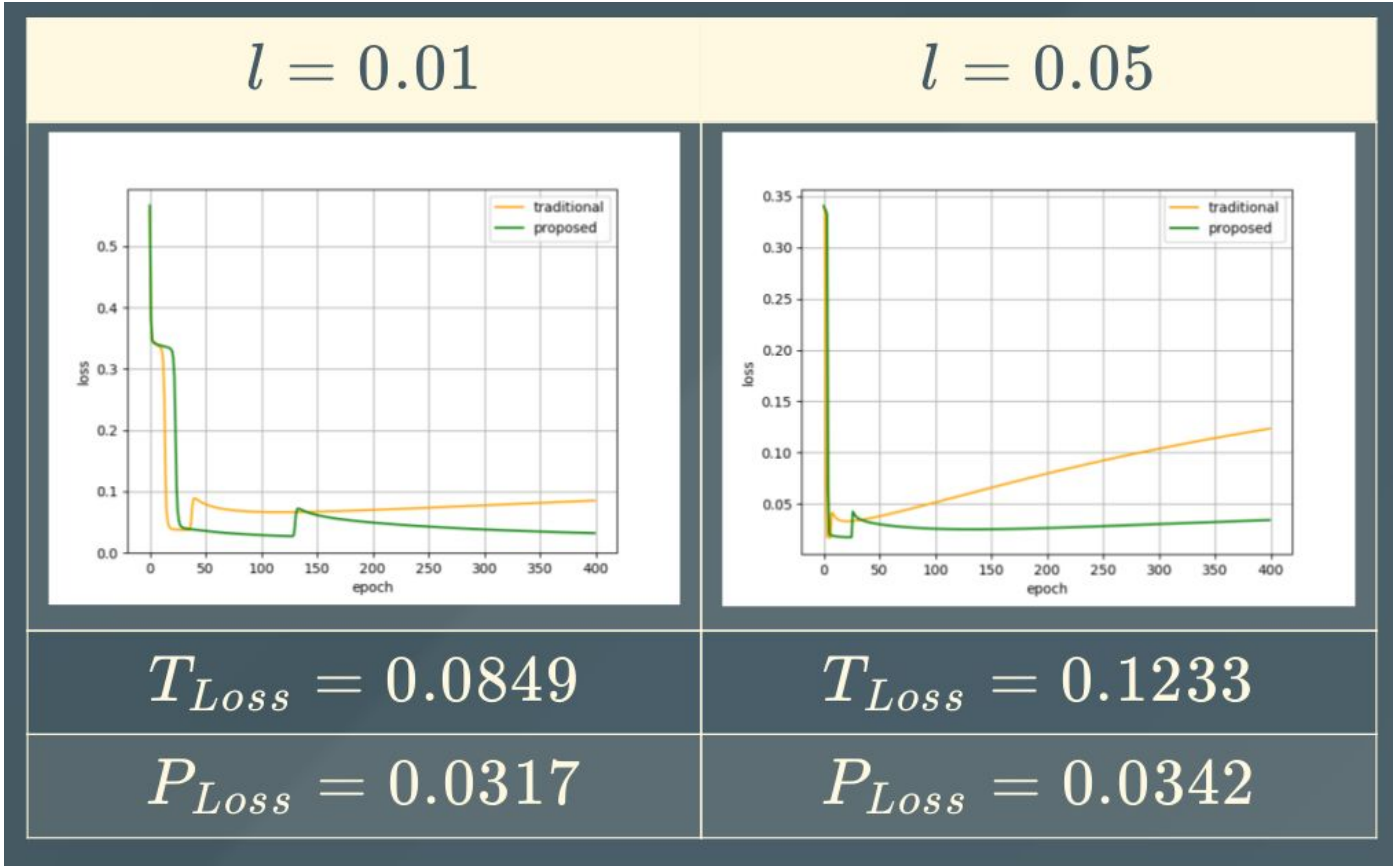


$$y_{\text{linear}} = wx + b$$

$$y_{\text{logistic}} = \frac{1}{1+e^{-(wx+b)}}$$

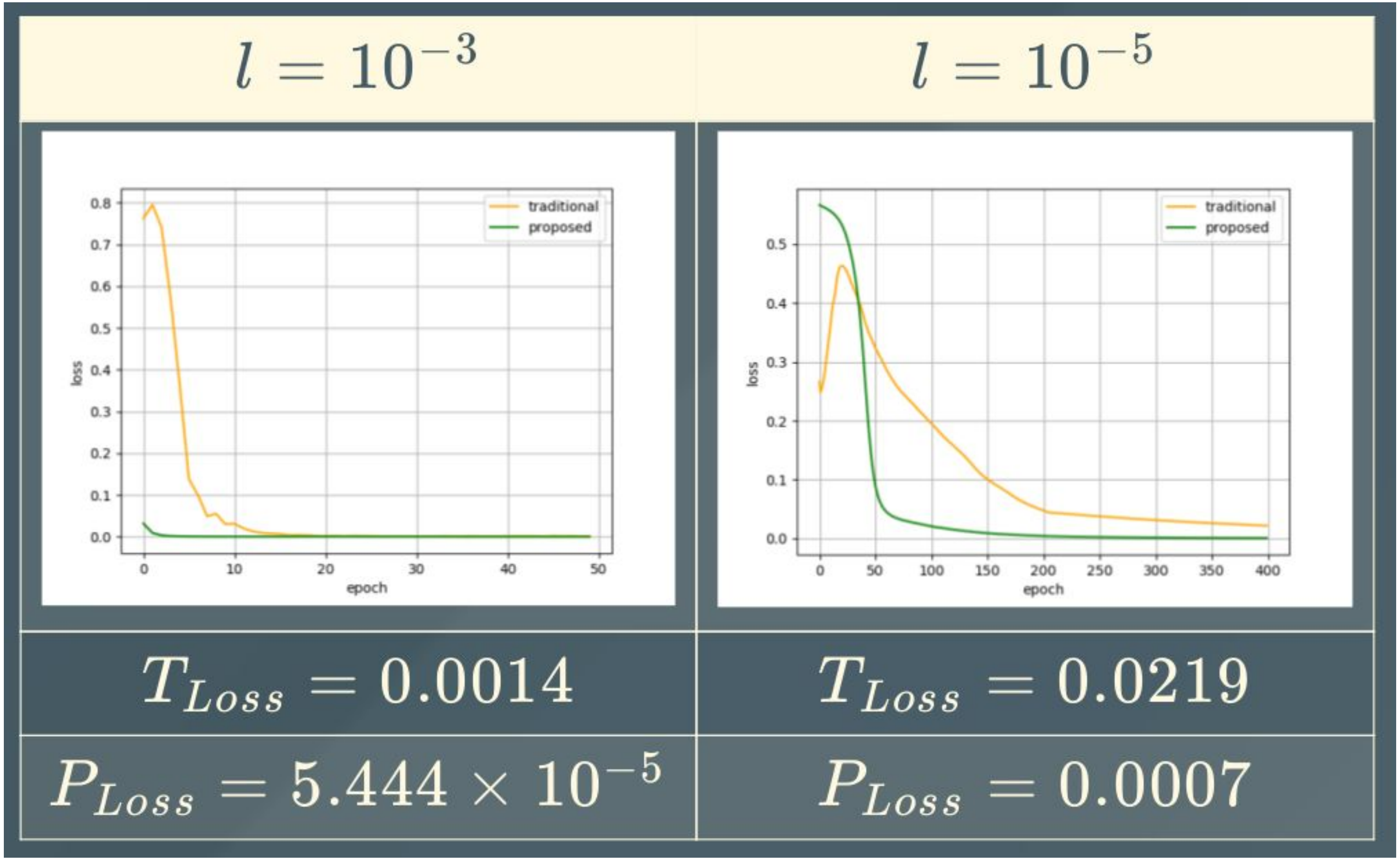
Ref: http://www.datacamp.com/

LOGISTIC REGRESSION



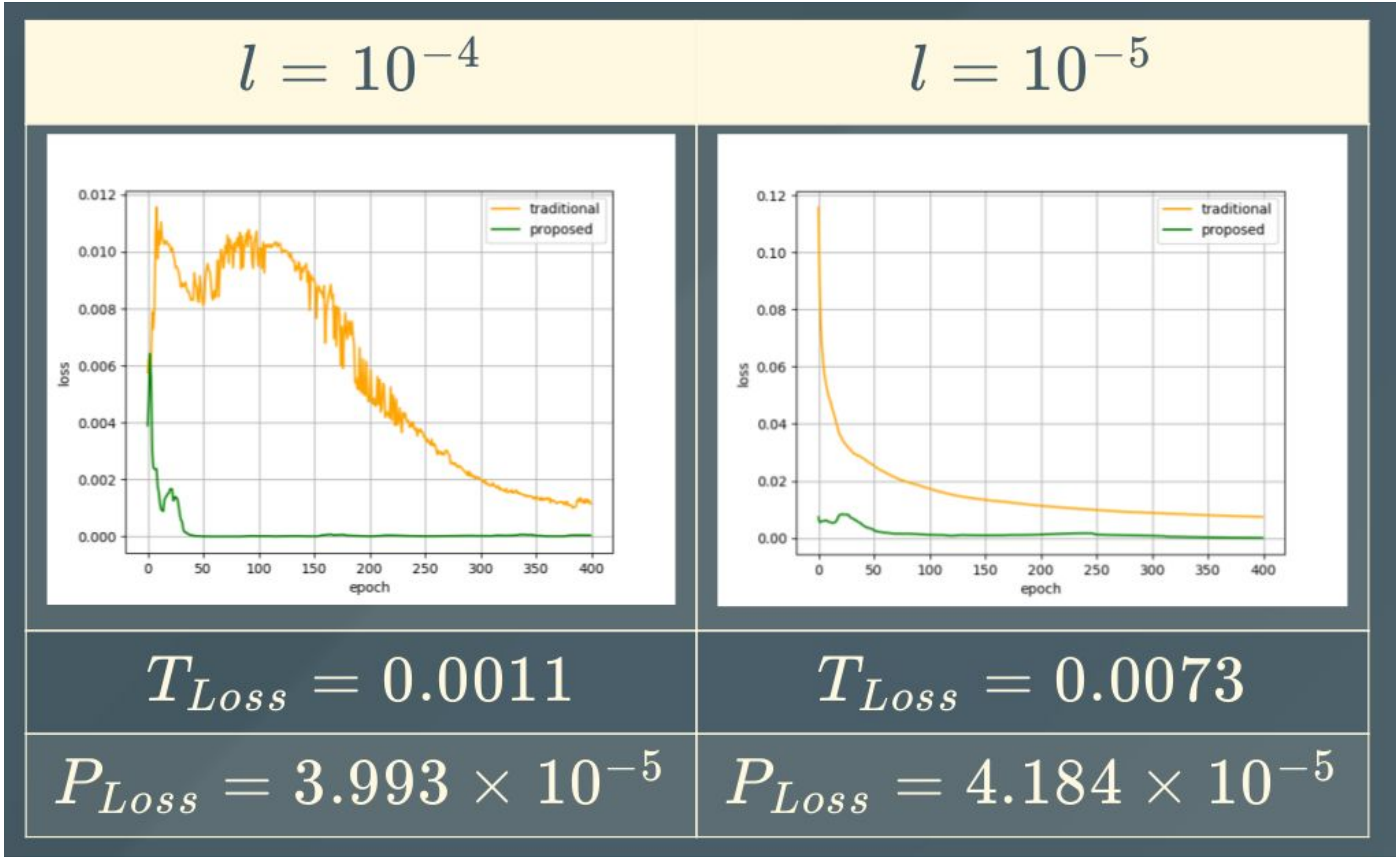
$l = 0.01$	$l = 0.05$
$T_{Loss} = 0.0849$	$T_{Loss} = 0.1233$
$P_{Loss} = 0.0317$	$P_{Loss} = 0.0342$

MULTI LAYER PERCEPTRON



$l = 10^{-3}$	$l = 10^{-5}$
$T_{Loss} = 0.0014$	$T_{Loss} = 0.0219$
$P_{Loss} = 5.444 \times 10^{-5}$	$P_{Loss} = 0.0007$

CNN



$l = 10^{-4}$	$l = 10^{-5}$
$T_{Loss} = 0.0011$	$T_{Loss} = 0.0073$
$P_{Loss} = 3.993 \times 10^{-5}$	$P_{Loss} = 4.184 \times 10^{-5}$

Thanks!

- Q & A
- Reach us at

  Email - [rewanth|nikhilj]@payatu.com

  Twitter - @Rewanth_Cool | @nikhilj_73