AI VIETNAM
All-in-One Course

Imbalanced Data

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2024

Outline

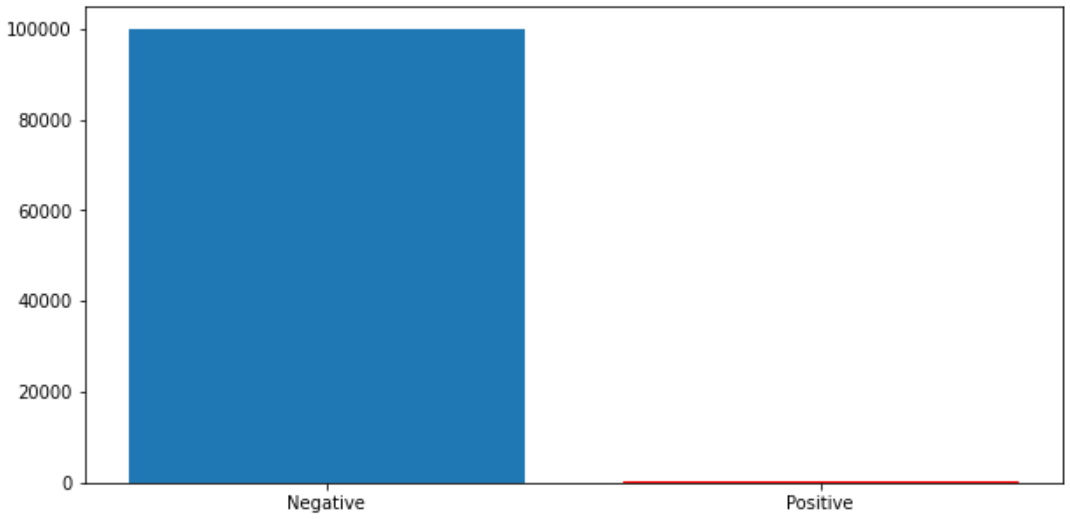
- Introduction
- Examples and Discussion
- Metrics
- Case Study

Introduction

❖ Imbalanced Data vs. Balanced Data

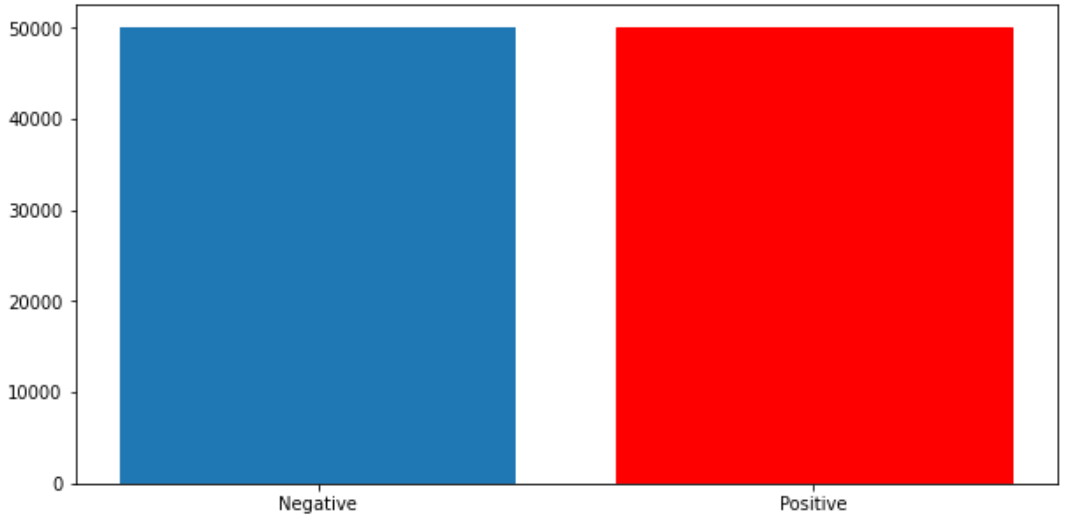
Imbalanced Data

Negative	100000
Positive	200



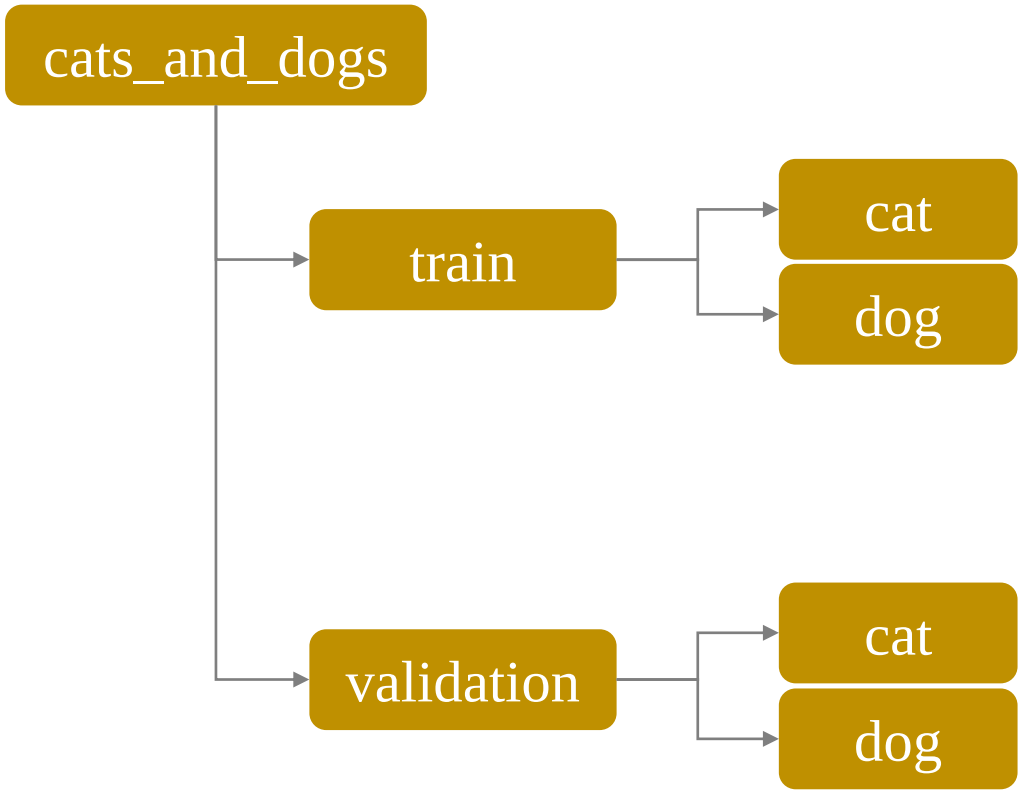
Balanced Data

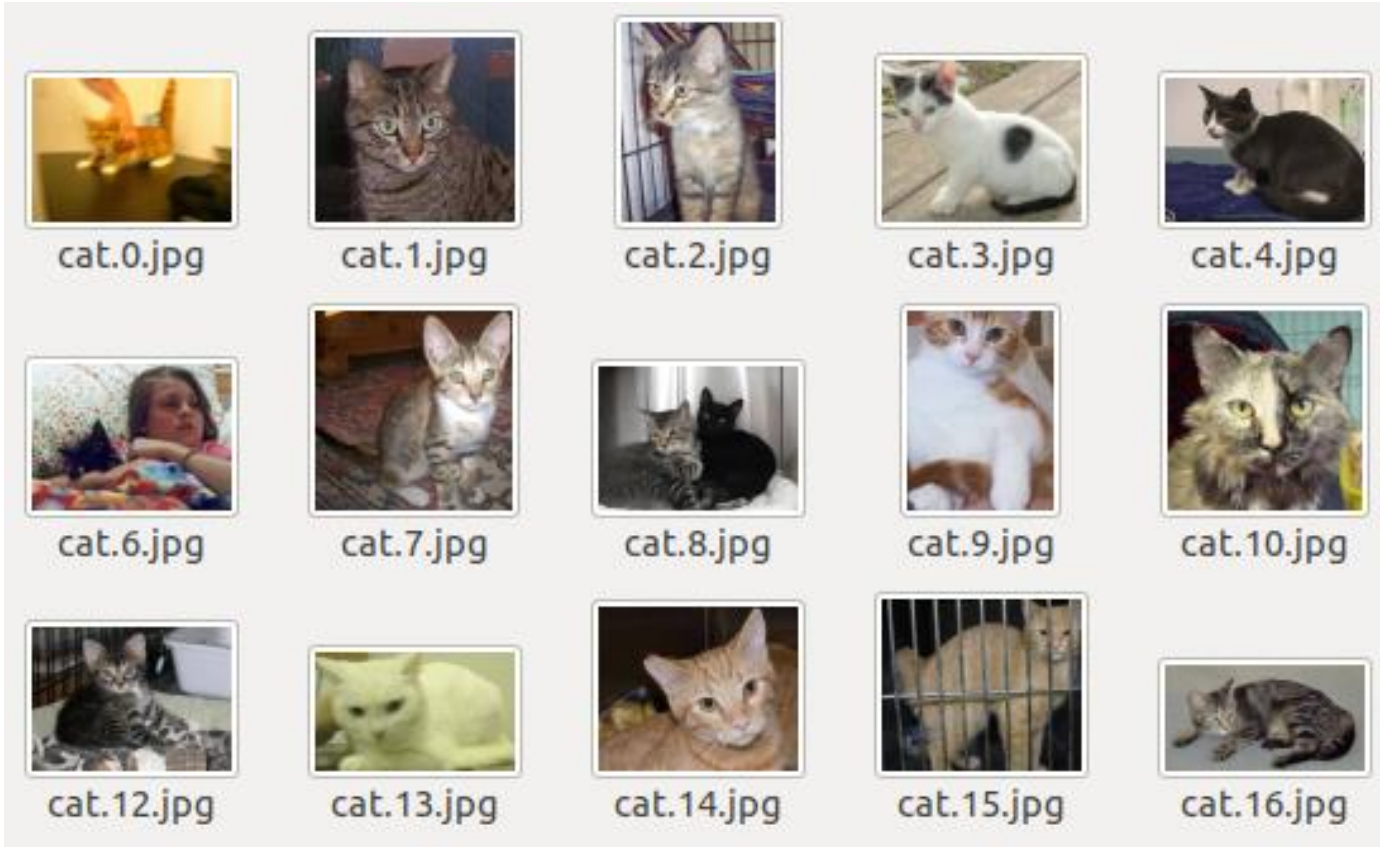
Negative	50100
Postive	50100

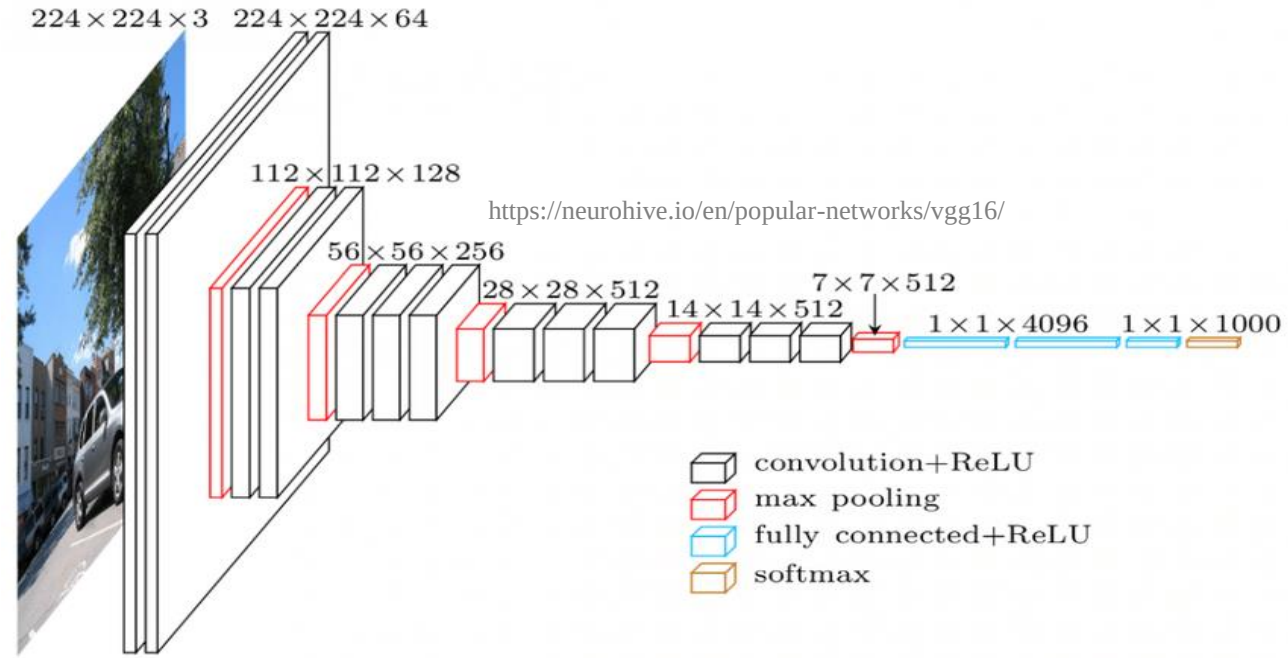


Introduction

❖ Cat-Dog dataset







https://neurohive.io/en/popular-networks/vgg16/

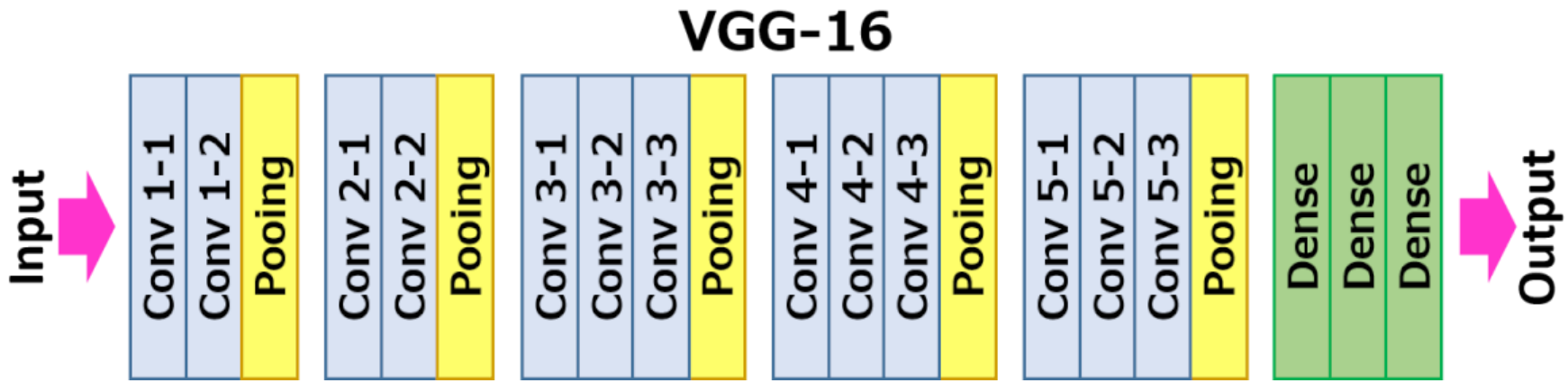
(3x3) Convolution padding='same' stride=1 + ReLU

(2x2) max pooling

Dense Layer + ReLU (4096)

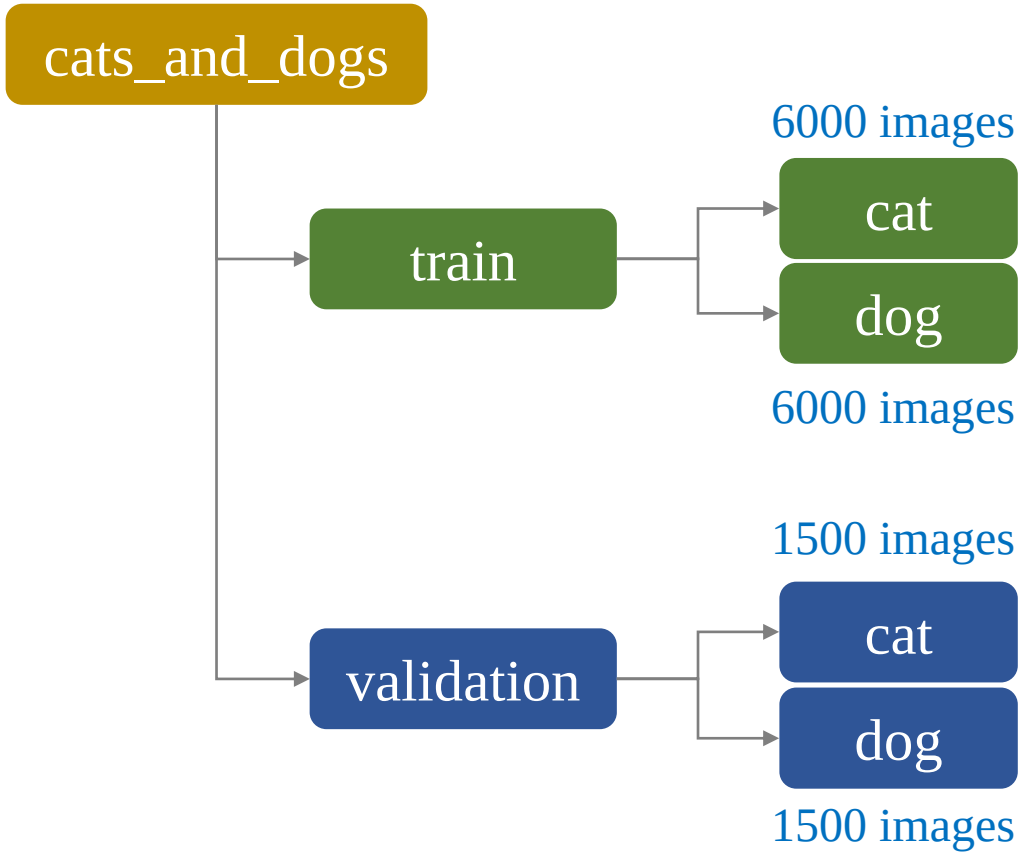
Dense Layer + Softmax (1000)

VGG16



approximately

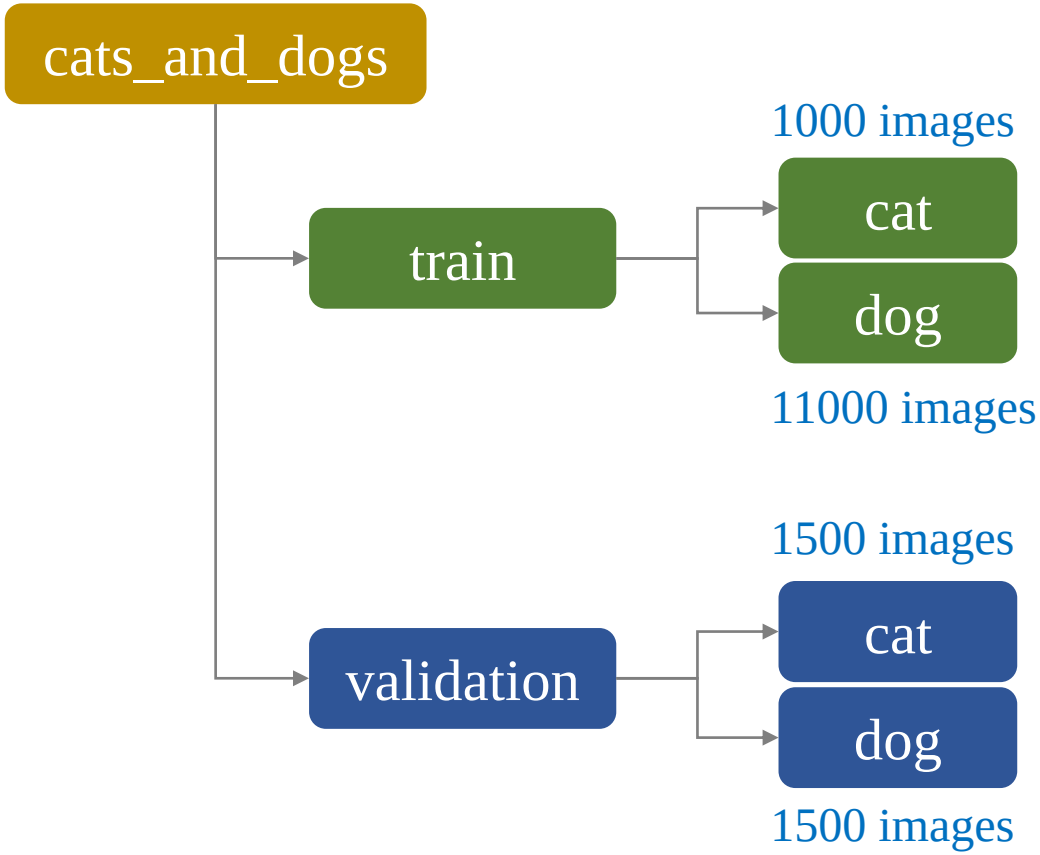
Introduction



Correct prediction

$\#_{cat} = 1245$

$\#_{dog} = 1236$



Correct prediction

$\#_{cat} = 1052$

$\#_{dog} = 1443$

Introduction

❖ Why?

Correct prediction (1)

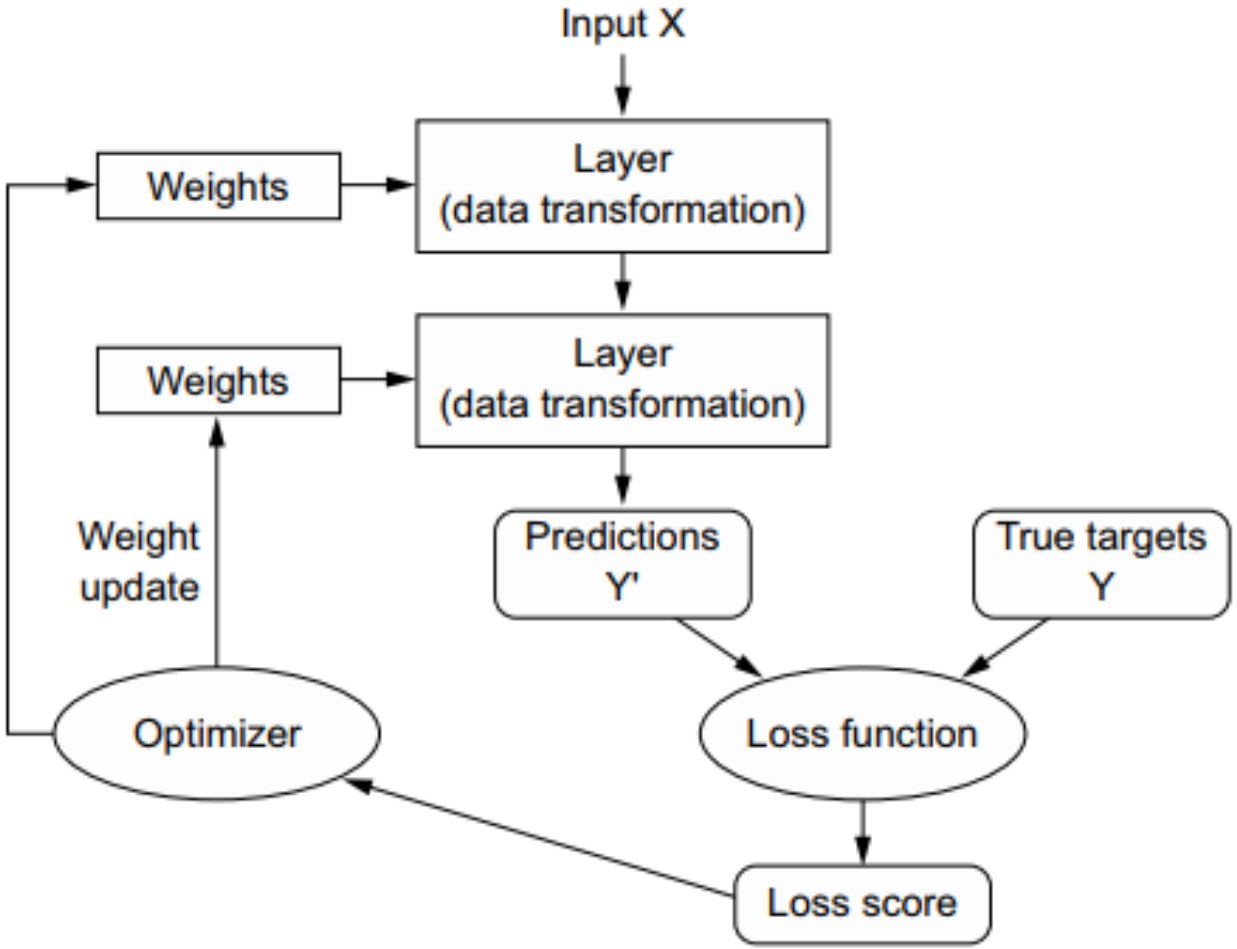
$\#_{cat} = 1245$

$\#_{dog} = 1236$

Correct prediction (2)

$\#_{cat} = 1052$

$\#_{dog} = 1443$



Outline

- Introduction
- Examples and Discussion
- Metrics
- Case Study

Imbalanced Data

❖ Lie/Truth classification

Feature Output Label

Input	Output	Label
…	0.3	0
…	0.8	0
…	0.7	0
…	0.4	0
…	0.6	1
…	0.8	1
…	0.9	1
…	0.2	1

if y = 0

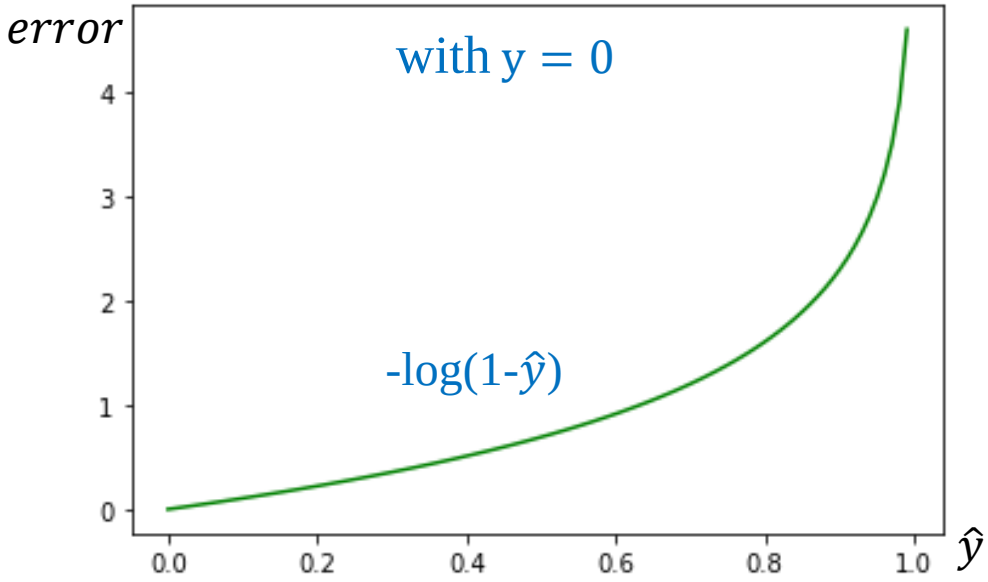
$$P(\hat{y}) = -\log(1-\hat{y})$$

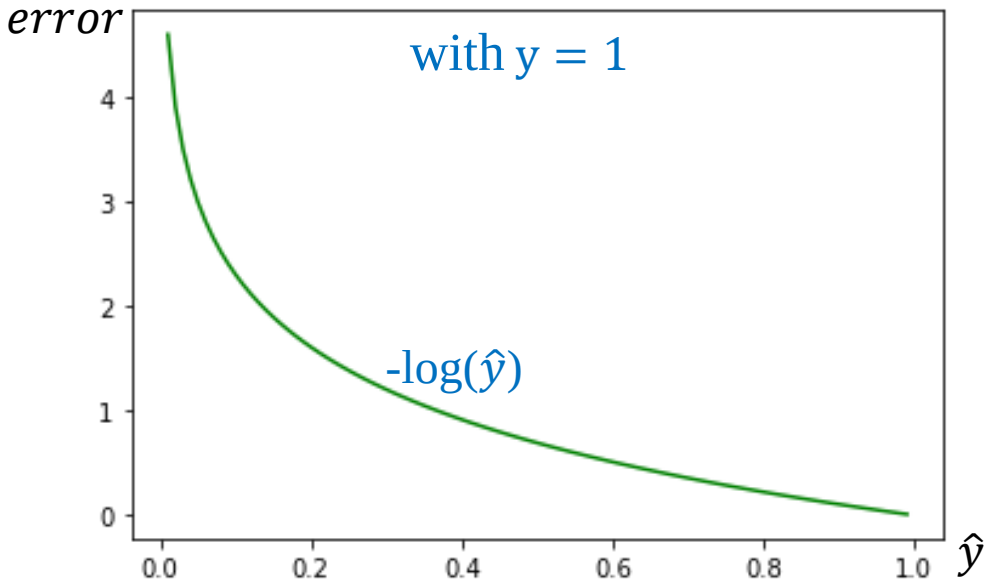
if y = 1

$$P(\hat{y}) = -\log(\hat{y})$$

Binary cross-entropy

$$L(.) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$





Imbalanced Data

❖ Lie/Truth classification

Feature Output Label

Input	Output	Label	Loss
…	0.3	0	0.356
…	0.8	0	1.609
…	0.7	0	1.203
…	0.4	0	0.511
…	0.6	1	0.511
…	0.8	1	0.223
…	0.9	1	0.105
…	0.2	1	1.609

if y = 0

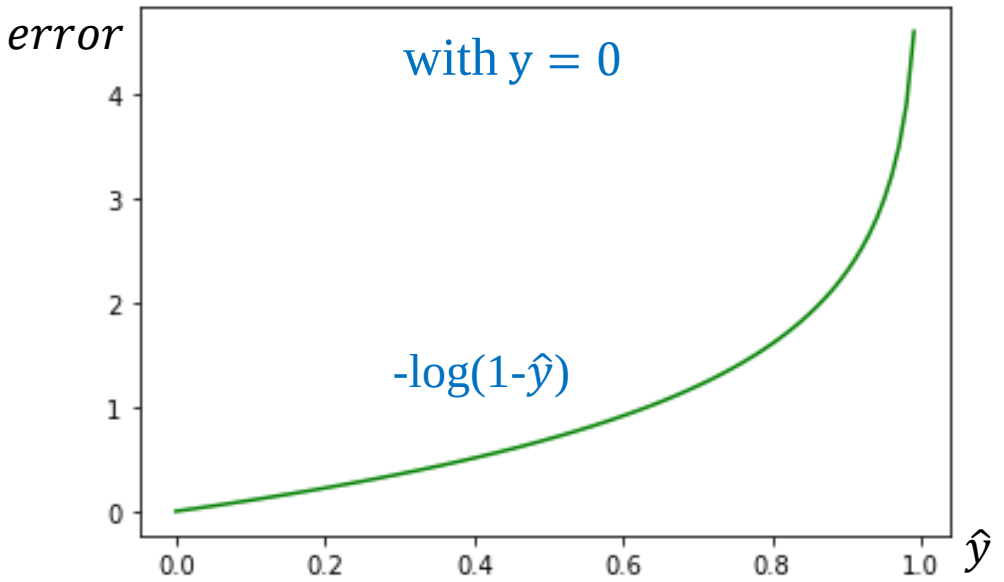
$$L(\hat{y}) = -\log(1-\hat{y})$$

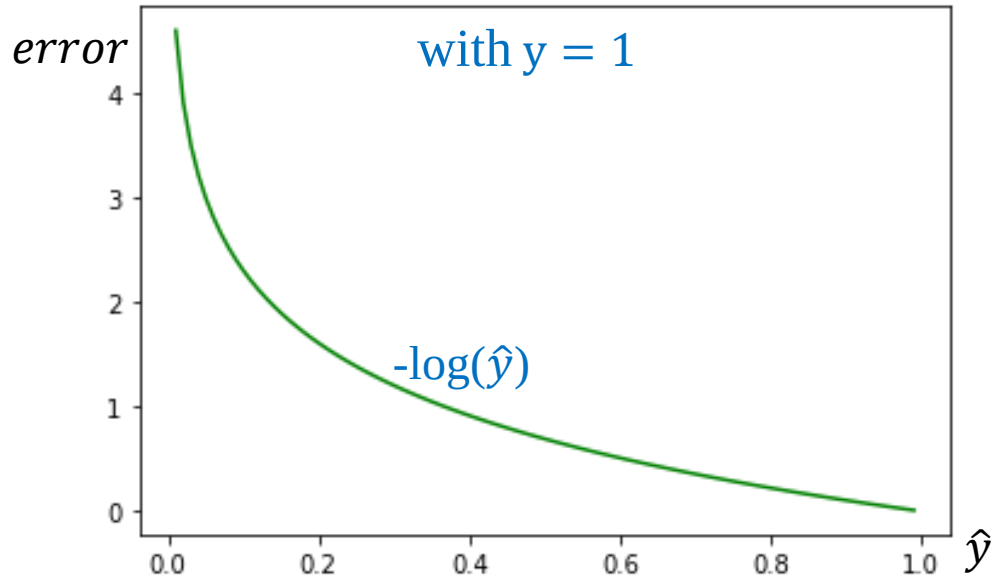
if y = 1

$$L(\hat{y}) = -\log(\hat{y})$$

Binary cross-entropy

$$L(y,\hat{y}) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$





Imbalanced Data

❖ Lie/Truth classification

Feature	Output	Label
Input	Output	Label
…	0.2	0
…	0.8	0
…	0.3	0
…	0.1	0
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1

After a period of time

if y = 0

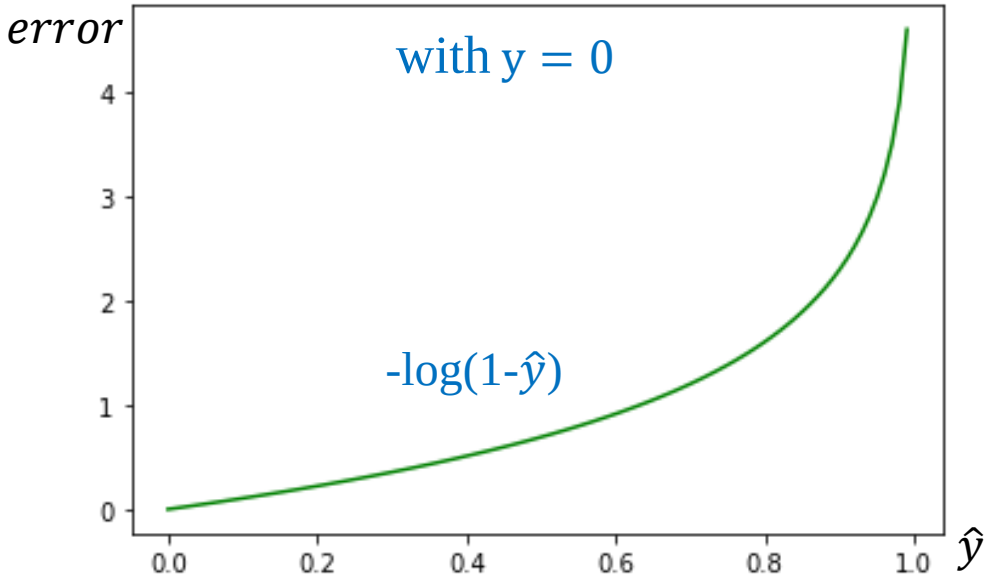
$$P(\hat{y}) = -\log(1-\hat{y})$$

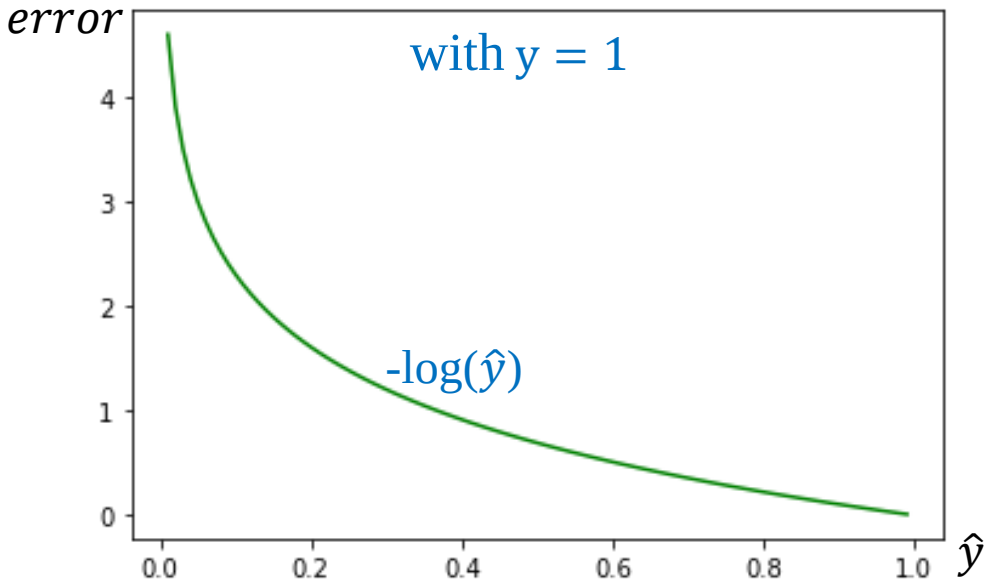
if y = 1

$$P(\hat{y}) = -\log(\hat{y})$$

Binary cross-entropy

$$L(.) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$





Imbalanced Data

❖ Lie/Truth classification

Feature	Output	Label
Input	Output	Label
…	0.2	0
…	0.7	0
…	0.7	1
…	0.8	1
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1
…	0.8	1
…	0.7	1

A special context

if y = 0

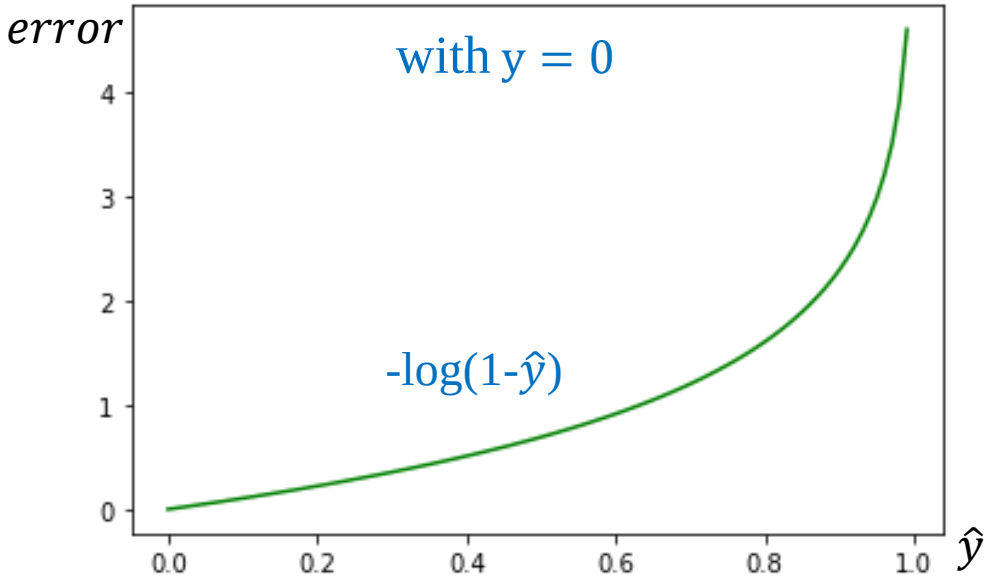
$$P(\hat{y}) = -\log(1-\hat{y})$$

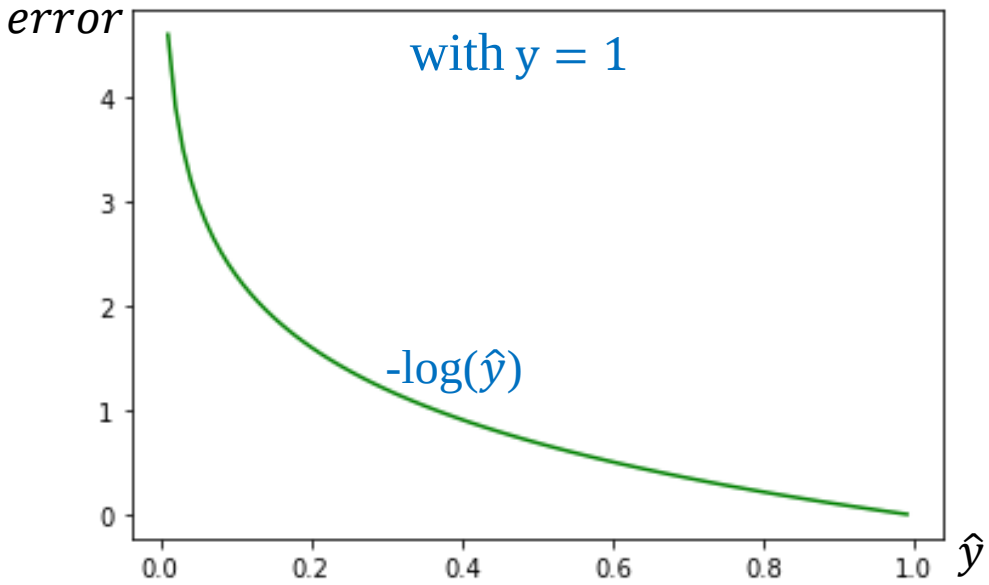
if y = 1

$$P(\hat{y}) = -\log(\hat{y})$$

Binary cross-entropy

$$L(.) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$





Imbalanced Data

❖ Lie/Truth classification

Feature	Output	Label
Input	Output	Label
…	0.7	0
…	0.8	1
…	0.7	1
…	0.8	1
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1
…	0.8	1
…	0.7	1

a more severe context!

$loss_{y=0} = 1.204$

$loss_{y=1} = 2.039$

if y = 0

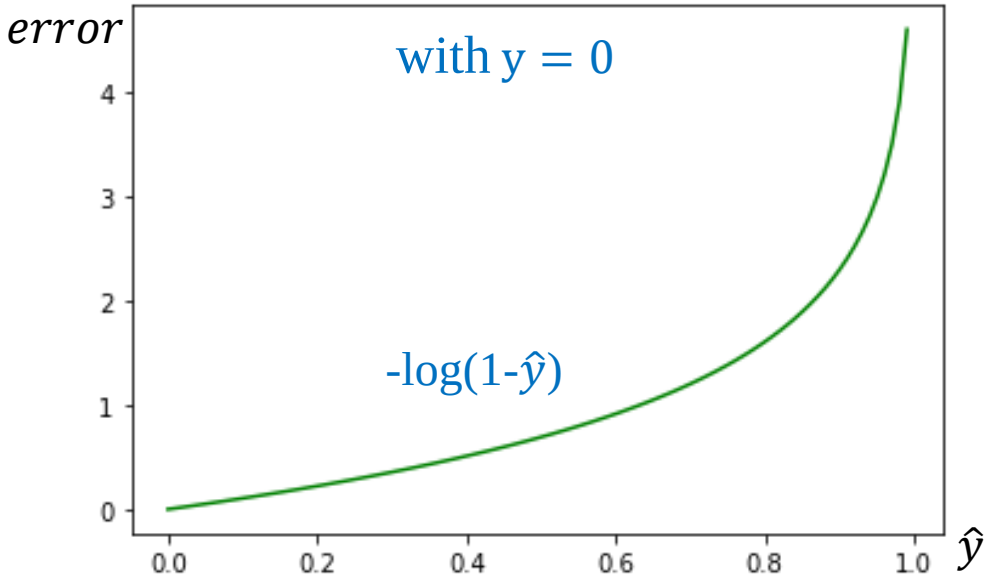
$$P(\hat{y}) = -\log(1-\hat{y})$$

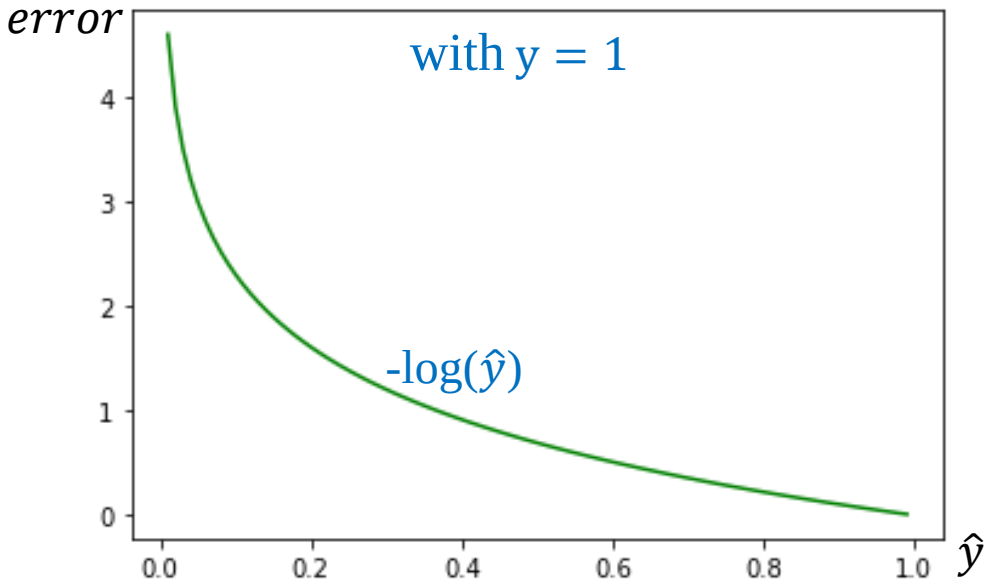
if y = 1

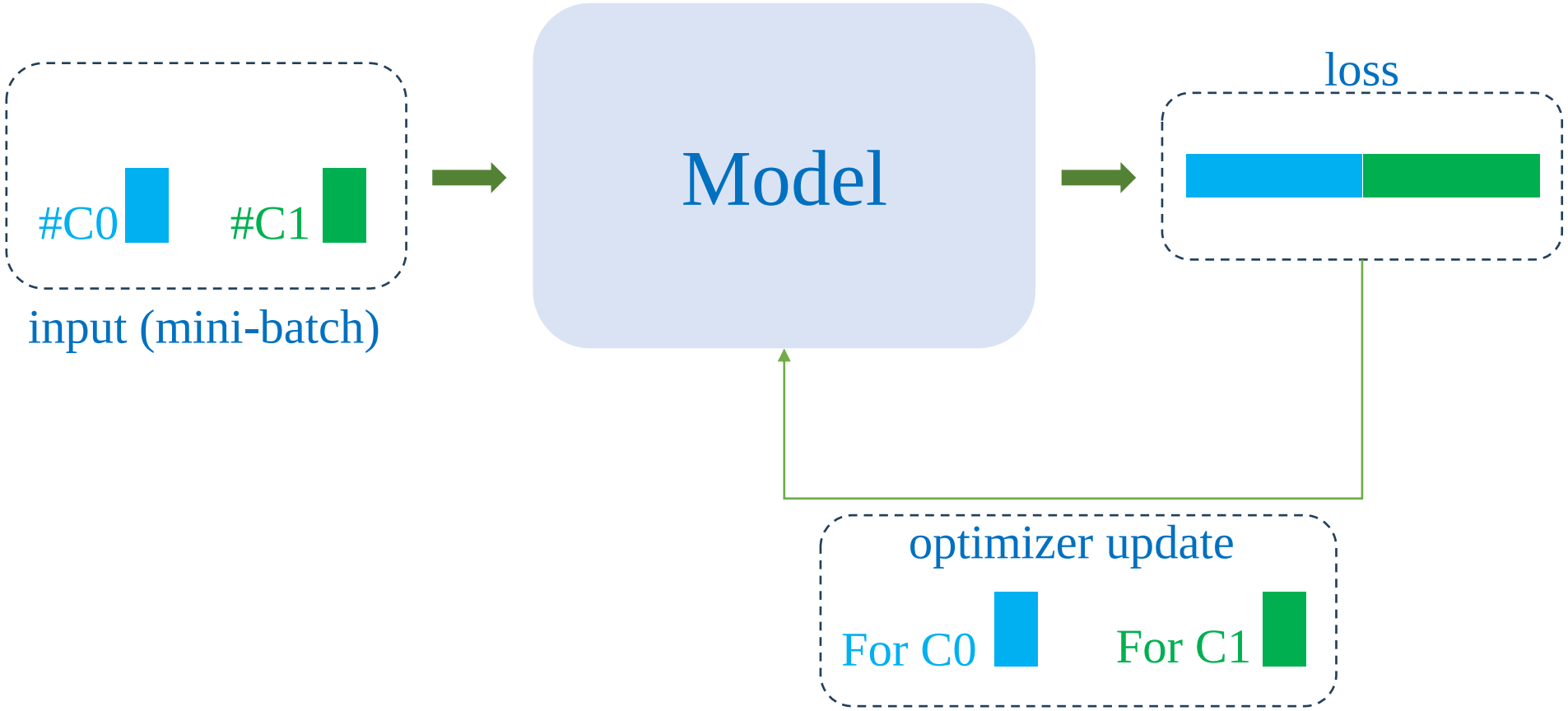
$$P(\hat{y}) = -\log(\hat{y})$$

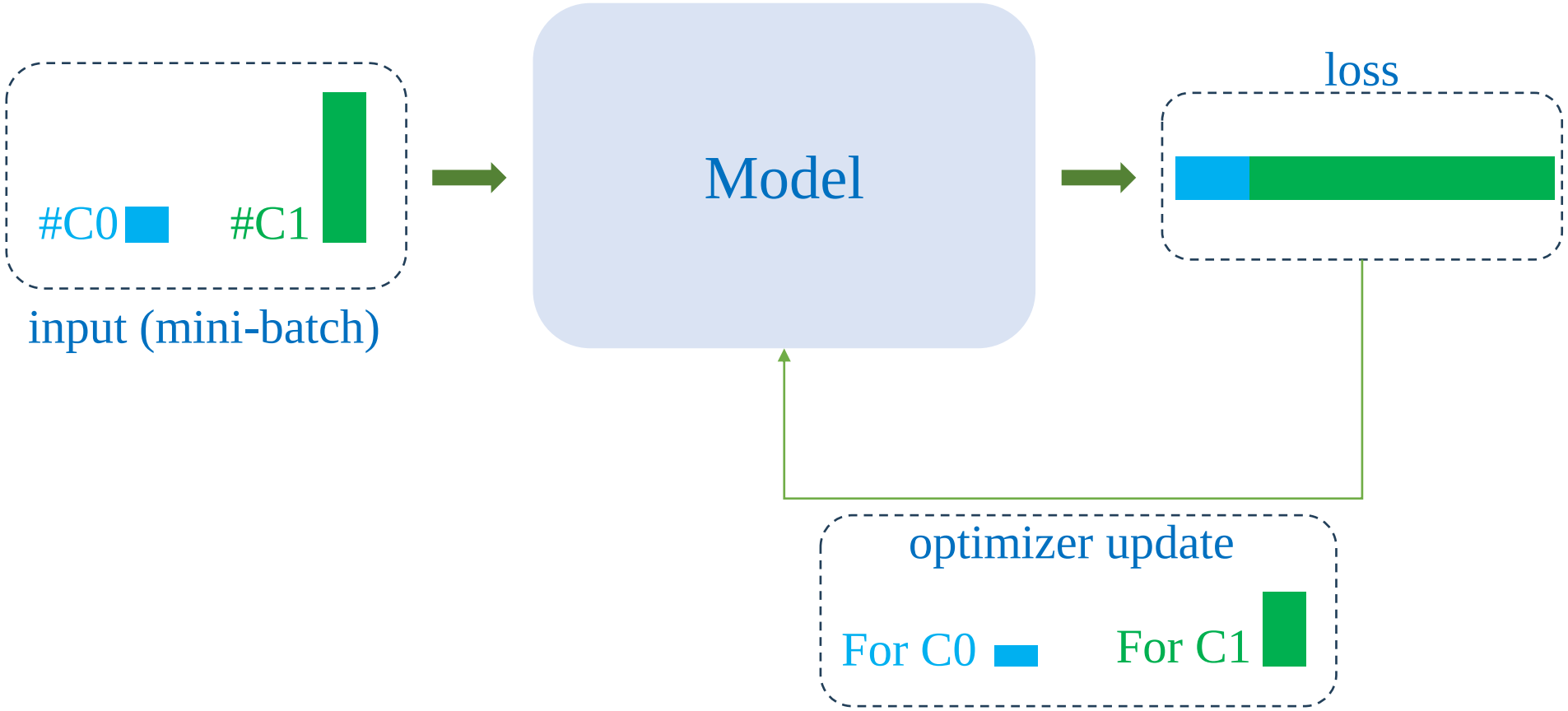
Binary cross-entropy

$$P = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$

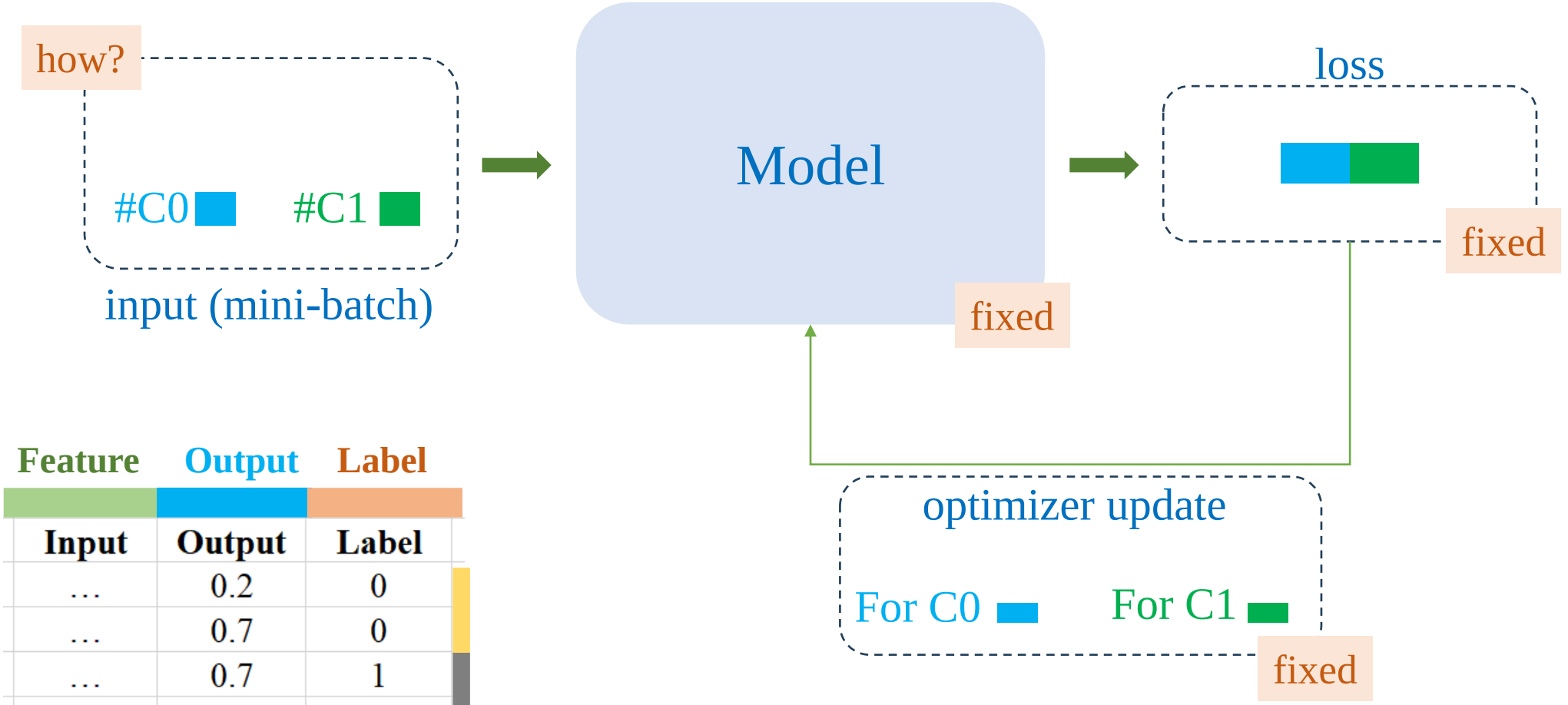





#C0
#C1
input (mini-batch)
Model
loss
optimizer update
For C0
For C1

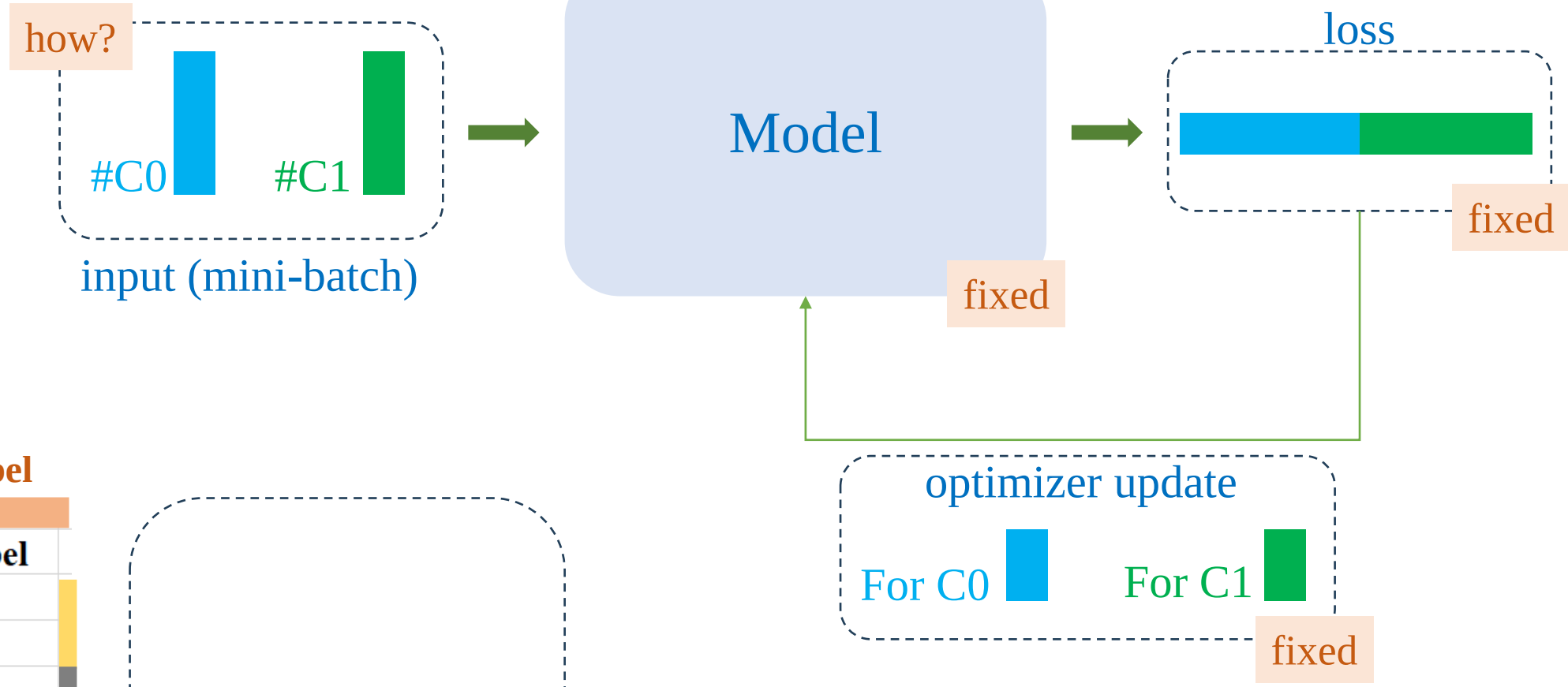

#C0
#C1
input (mini-batch)
Model
loss
optimizer update
For C0
For C1

solution 1



Feature	Output	Label
Input	Output	Label
…	0.2	0
…	0.7	0
…	0.7	1
…	0.8	1
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1
…	0.8	1
…	0.7	1

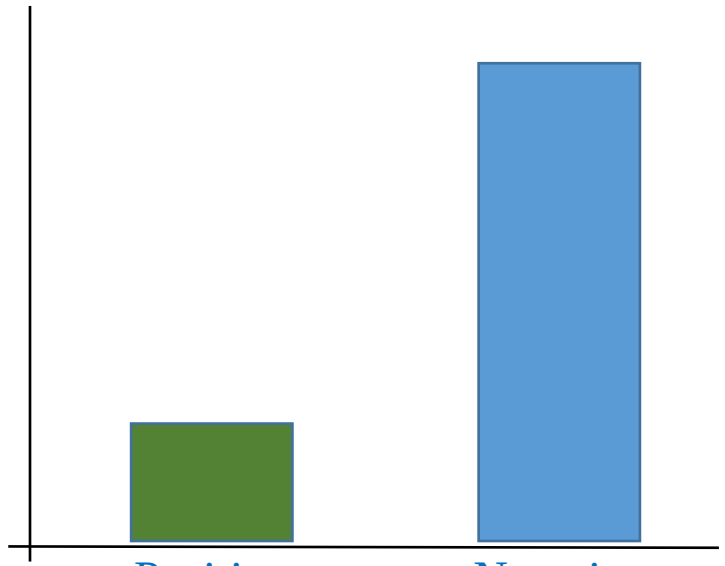
solution 2



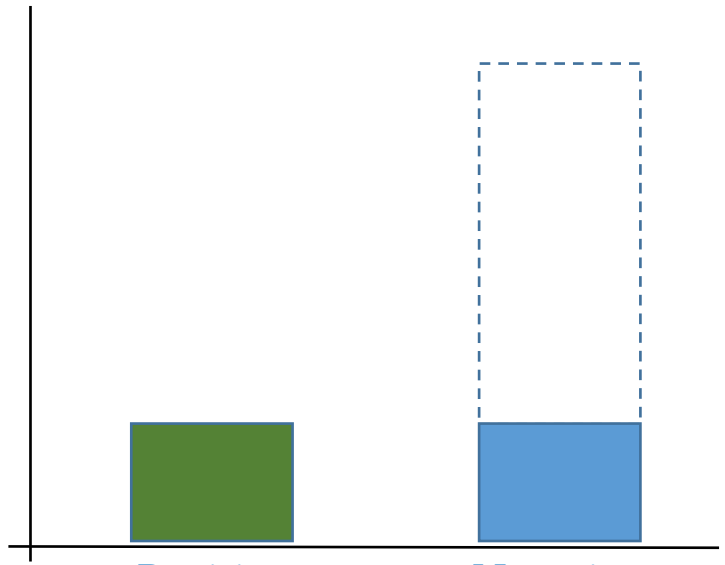
Feature	Output	Label
Input	Output	Label
…	0.2	0
…	0.7	0
…	0.7	1
…	0.8	1
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1
…	0.8	1
…	0.7	1

Imbalanced Data

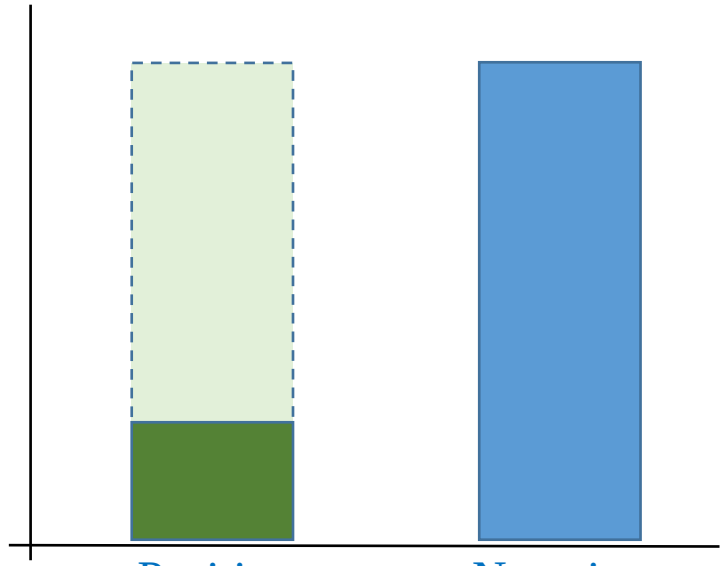
❖ Approach 1: Data manipulation



Original Data

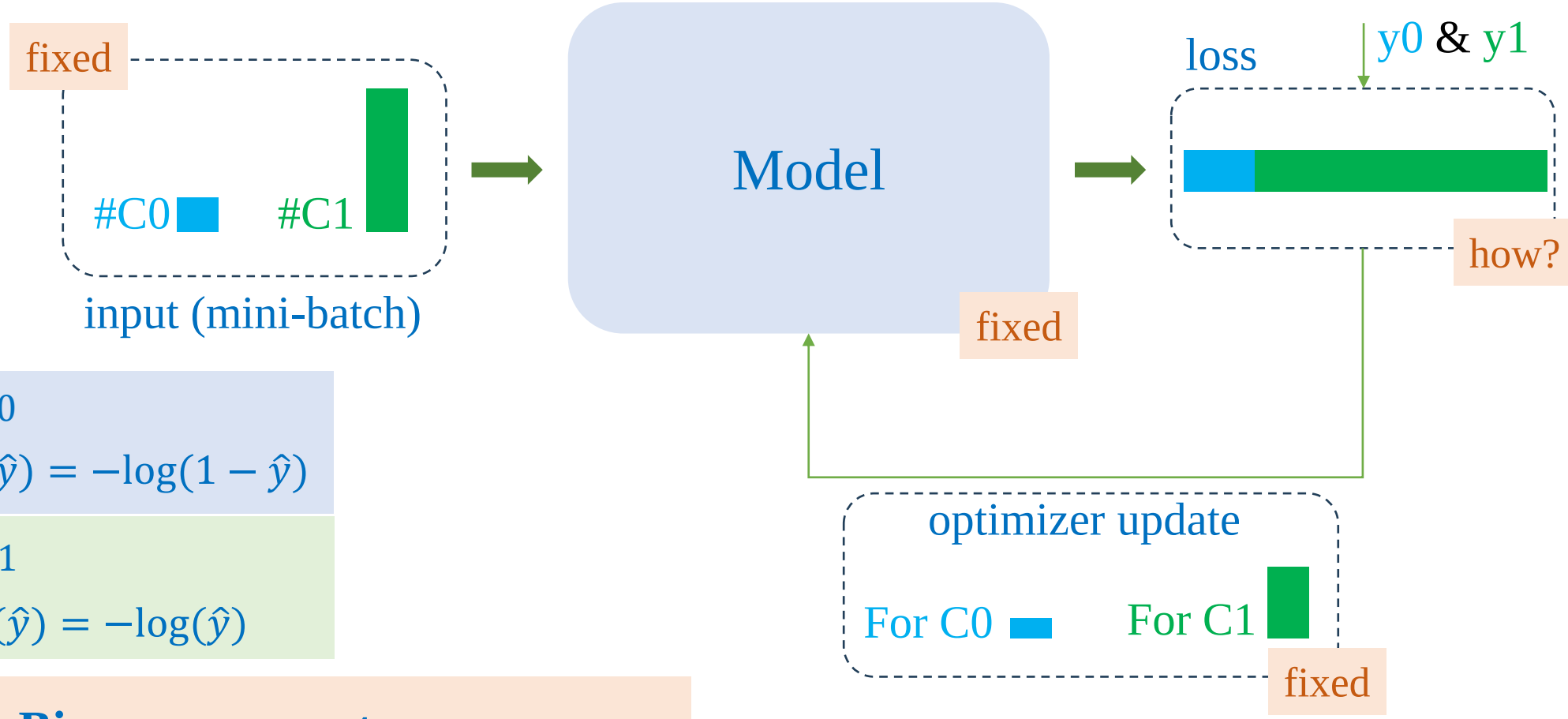


Undersampling Data



Oversampling Data

solution 3



if y = 0

$$L(\hat{y}) = -\log(1-\hat{y})$$

if y = 1

$$L(\hat{y}) = -\log(\hat{y})$$

Binary cross-entropy

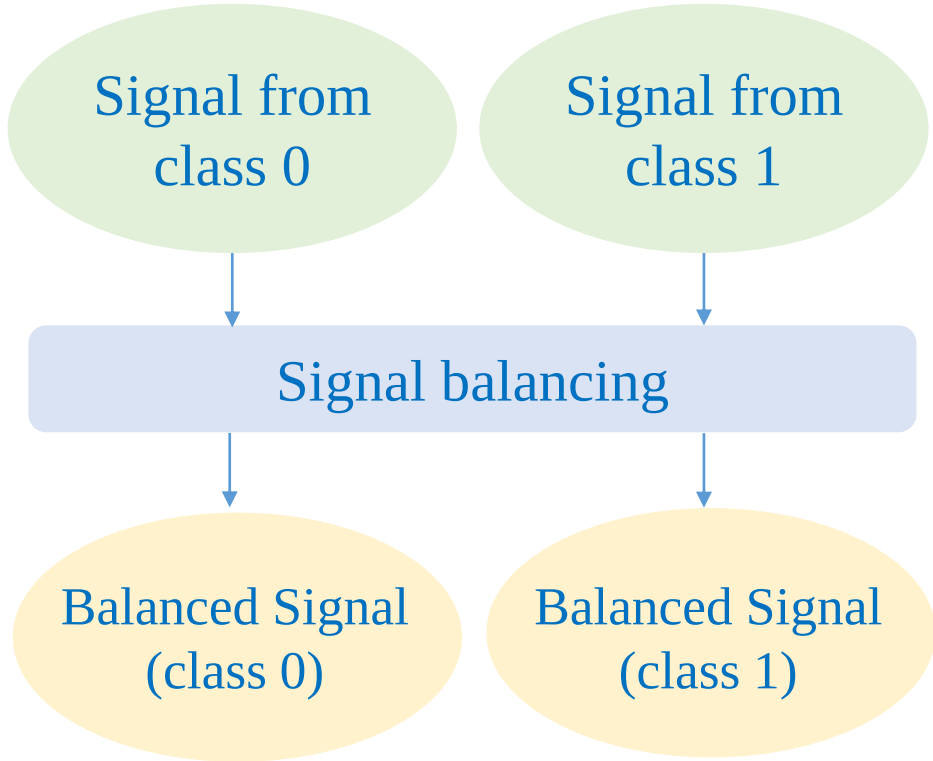
$$L(.) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$$

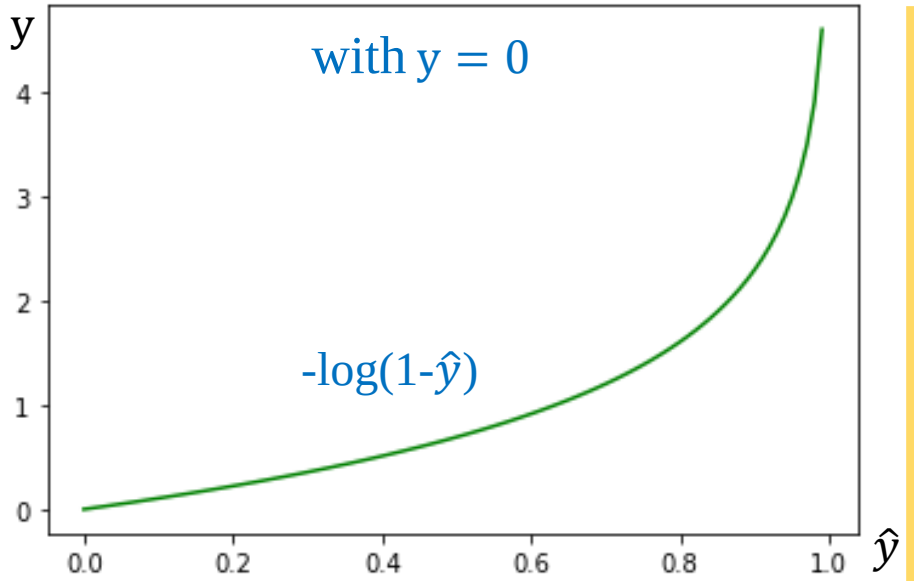
Imbalanced Data

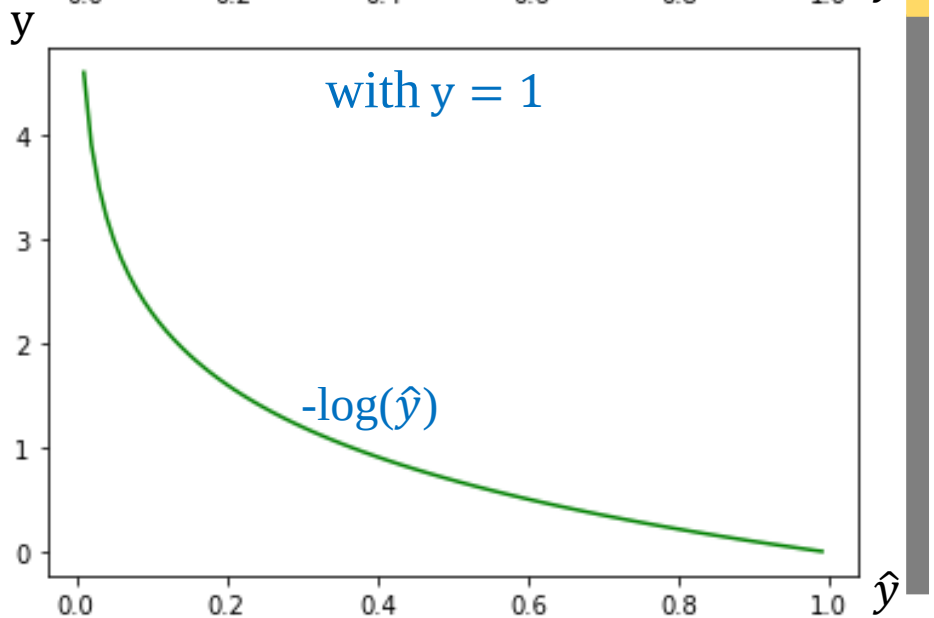
❖ Lie/Truth classification

Feature Output Label

Input	Output	Label
…	0.7	0
…	0.8	1
…	0.7	1
…	0.8	1
…	0.8	1
…	0.8	1
…	0.9	1
…	0.9	1
…	0.8	1
…	0.7	1





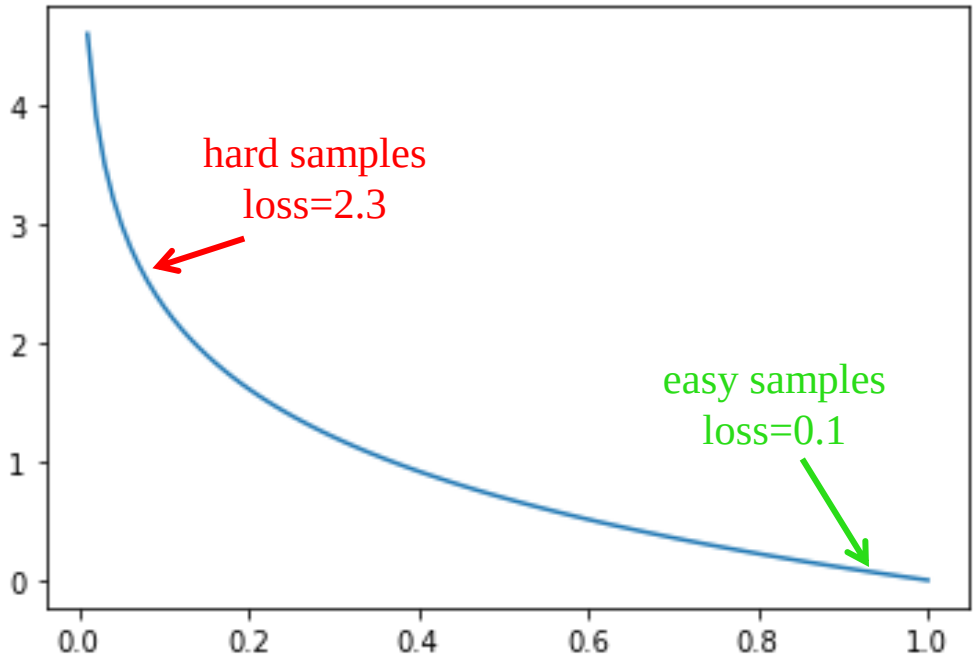


Class-weighted Binary cross-entropy

$$L(.) = -\alpha_1 y \log\hat{y} - \alpha_2 (1-y)\log(1-\hat{y})$$

Imbalanced Data

❖ Imbalance data



Easy samples loss = 100000*0.1 = 10000

Hard samples loss = 100*2.3 = 230

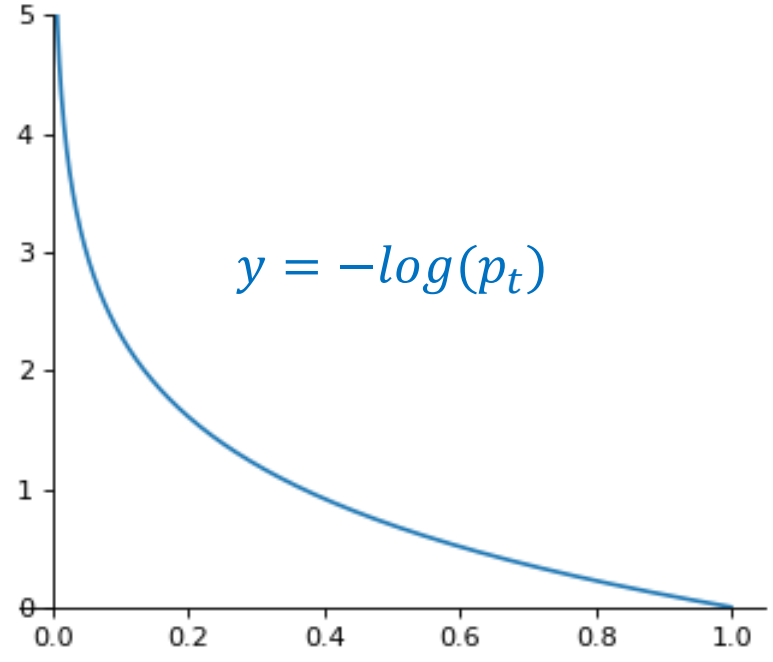
Loss = Easy samples loss + Hard samples loss

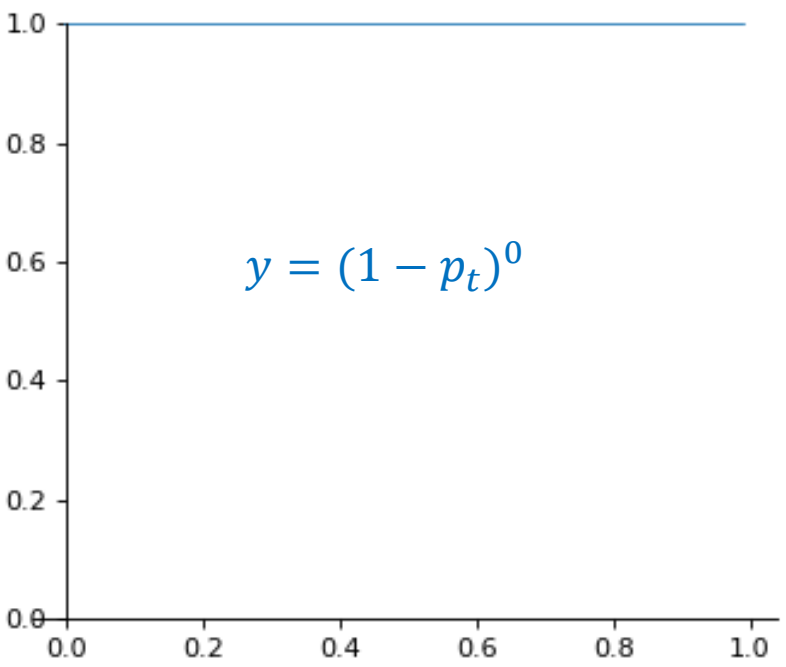
Easy samples loss : Hard samples loss = 10000:230 ≈ 43

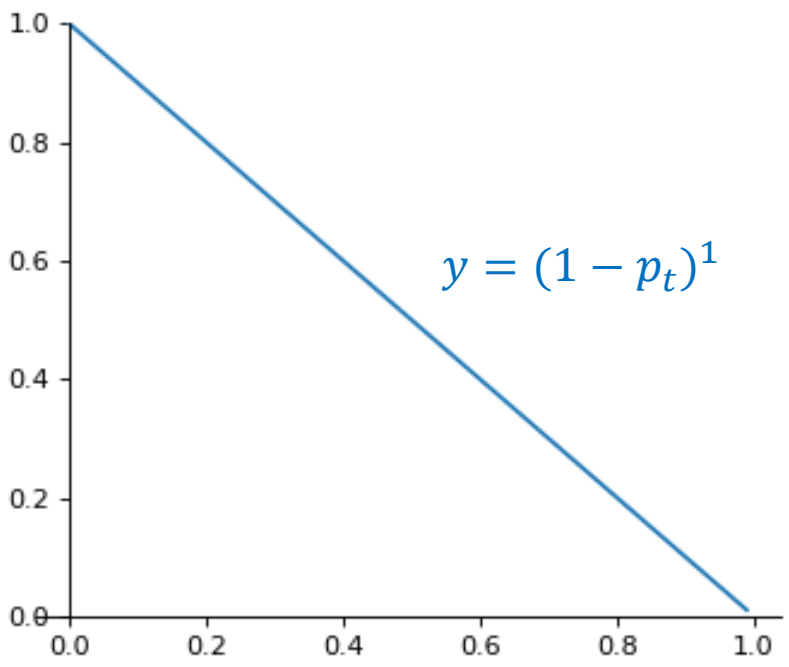
BCE không tốt cho trường hợp data bị imbalance nặng

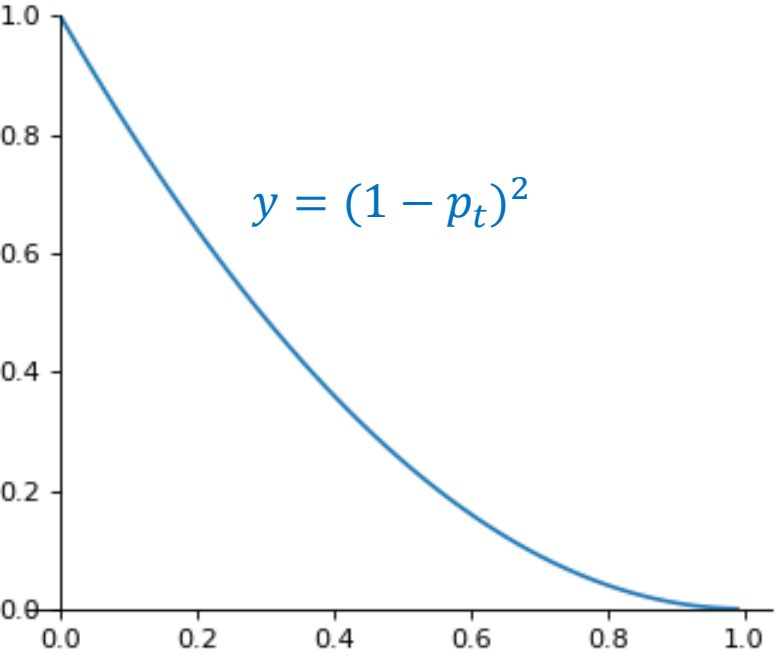
Imbalance Case:
- 100000 easy samples vs 100 hard samples

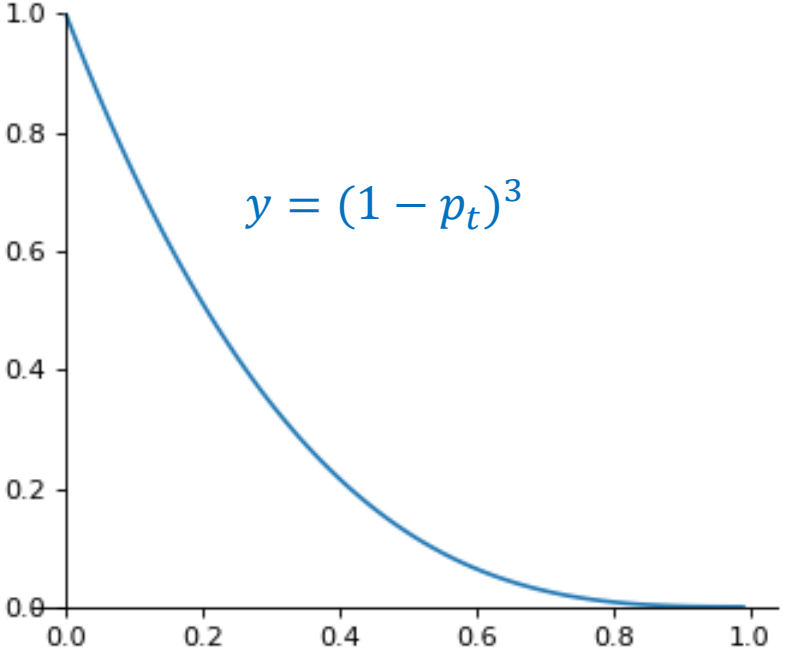
How to solve it!!!

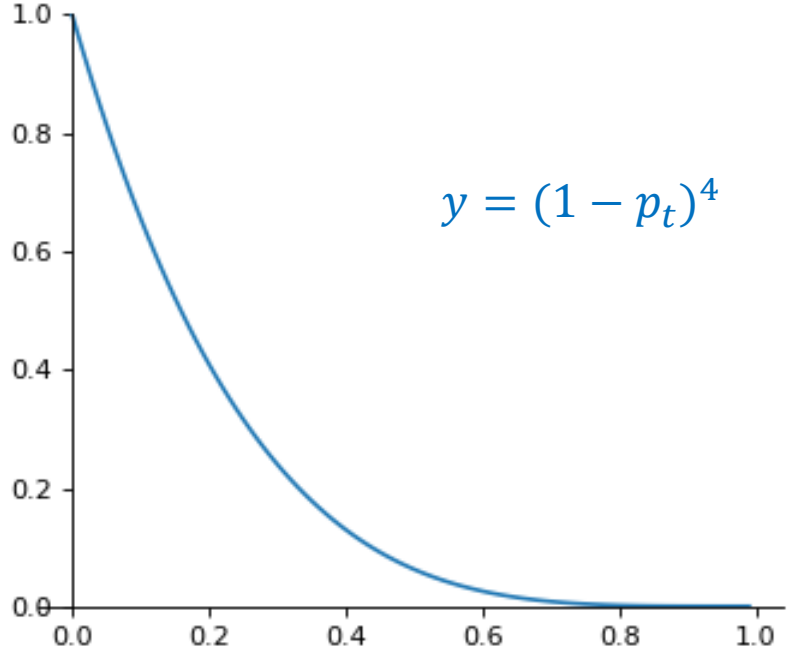

$y = -log(p_t)$


$y = (1 - p_t)^0$

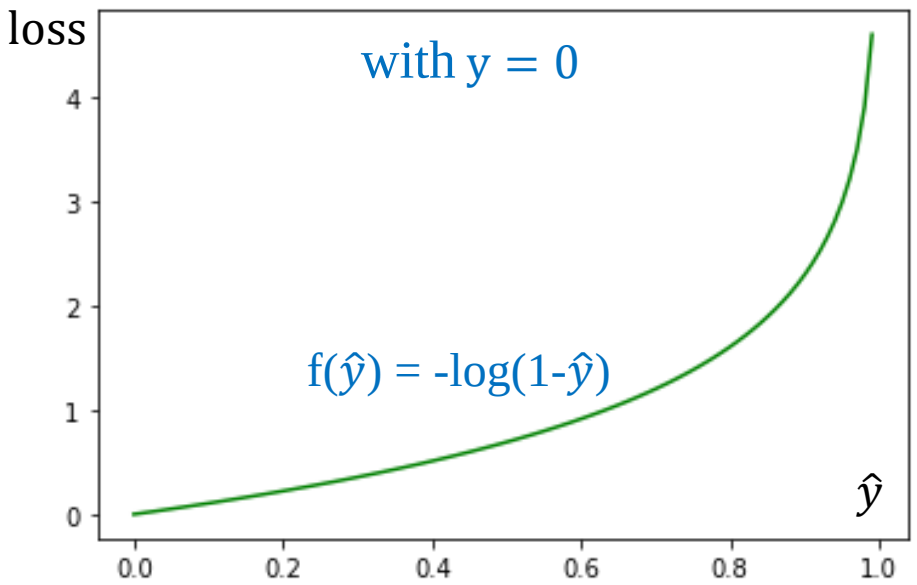

$y = (1 - p_t)^1$


$y = (1 - p_t)^2$


$y = (1 - p_t)^3$


$y = (1 - p_t)^4$

Designing a Function

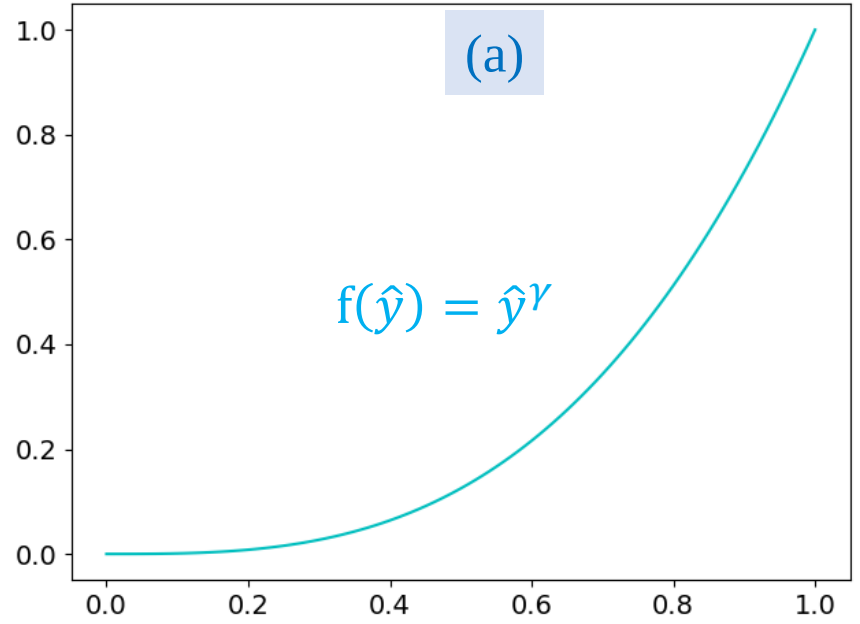


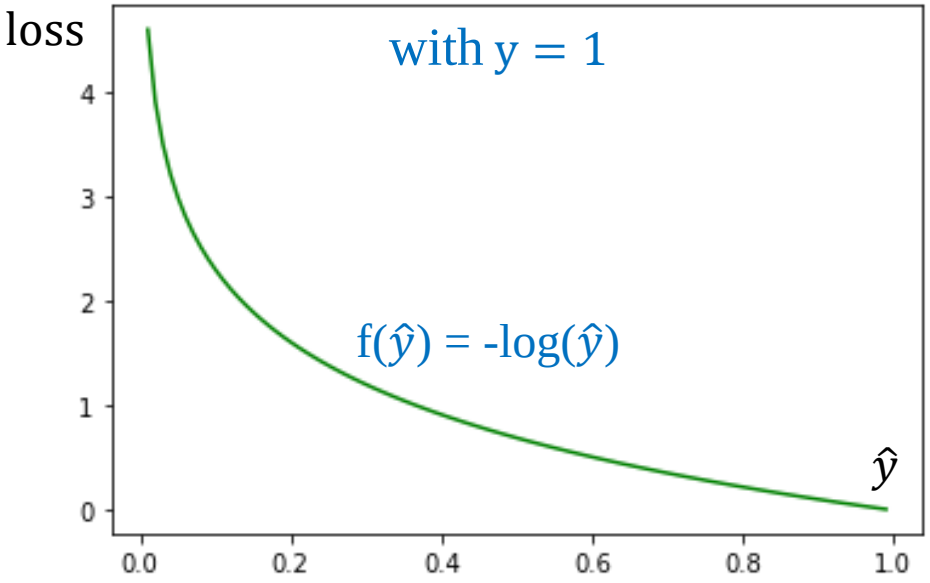
Given $0 \leq k \leq 1$

if $f(\hat{y}) * k$

where k approaches 1

→ $f(\hat{y}) * k$ reduces slightly

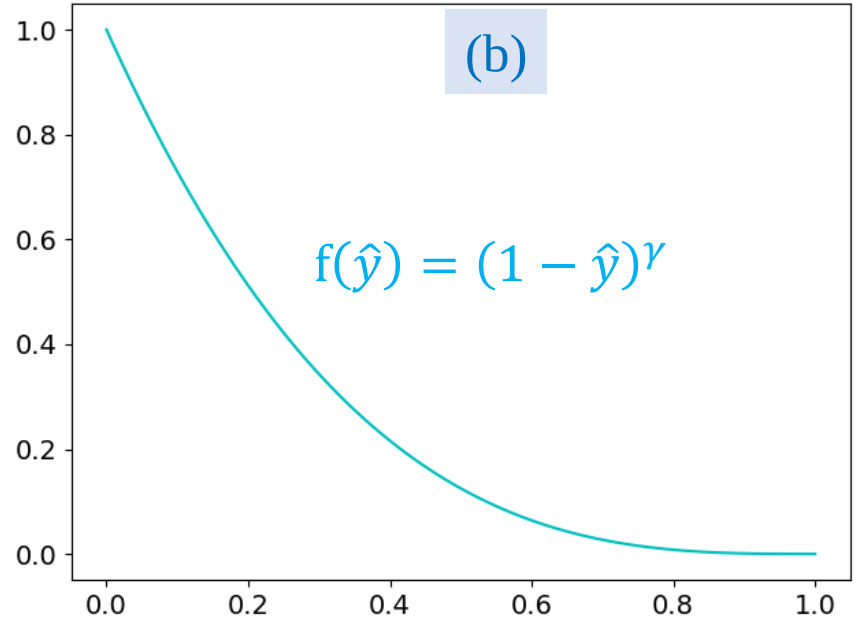




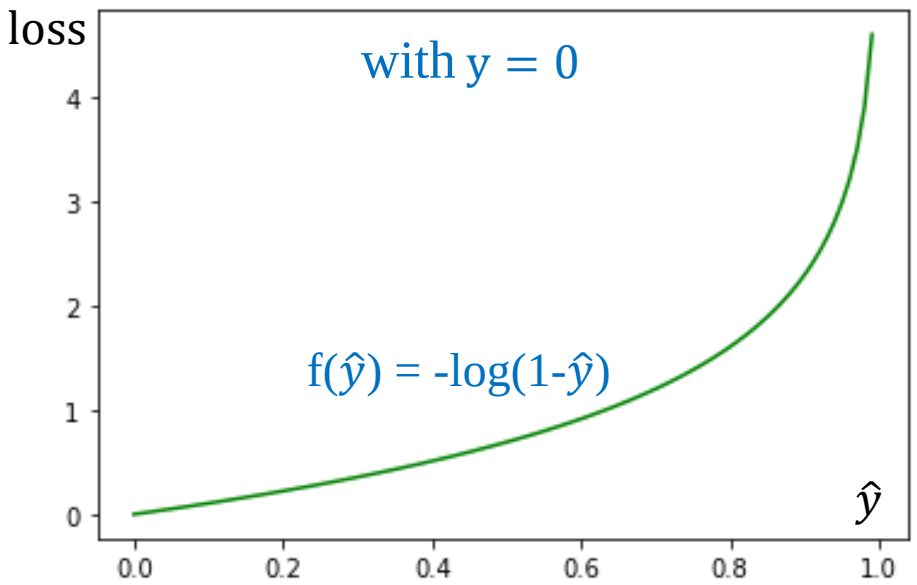
if $f(\hat{y}) * k$

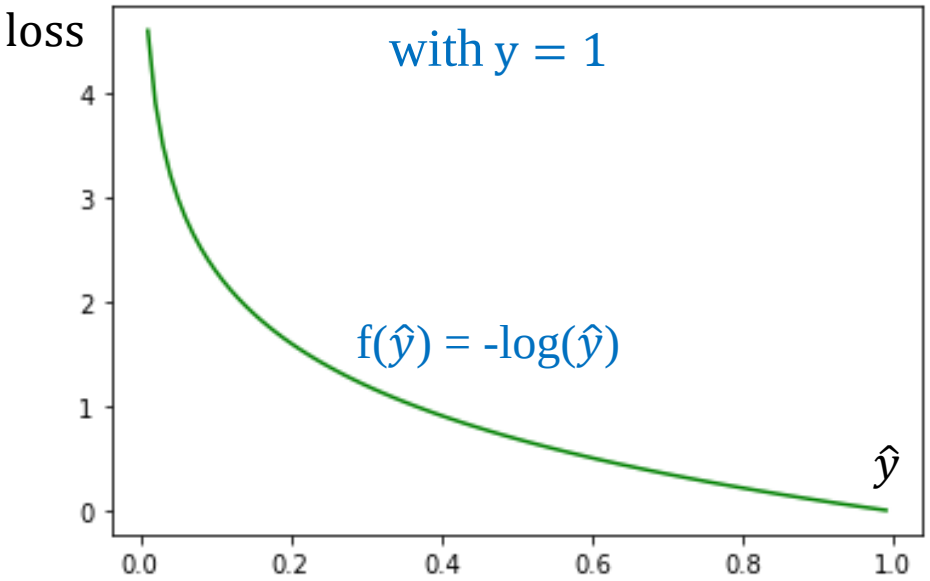
where k approaches 0

→ $f(\hat{y}) * k$ reduces significantly



Designing a Function





Given $0 \leq k \leq 1$

if $f(\hat{y}) * k$

where k approaches 1

→ $f(\hat{y}) * k$ reduces slightly

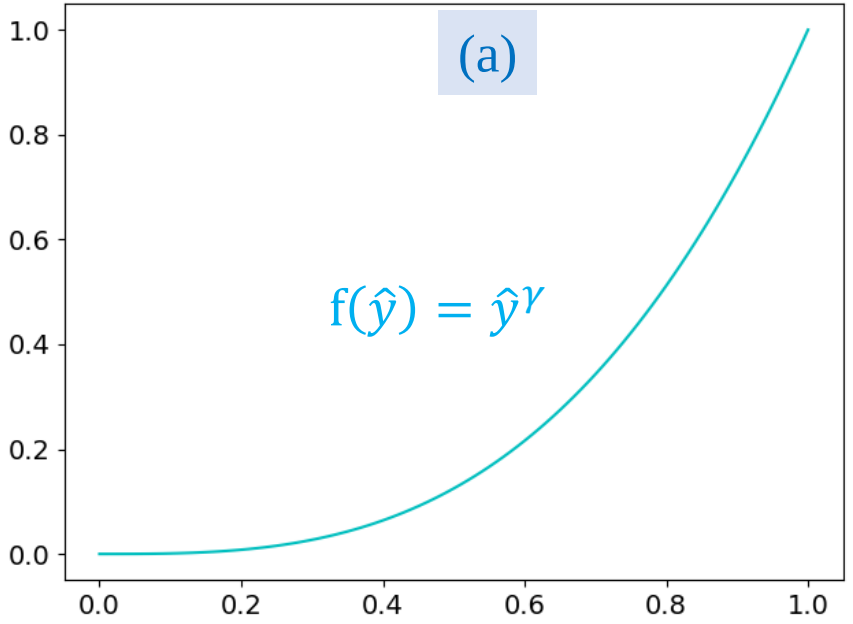
if $f(\hat{y}) * k$

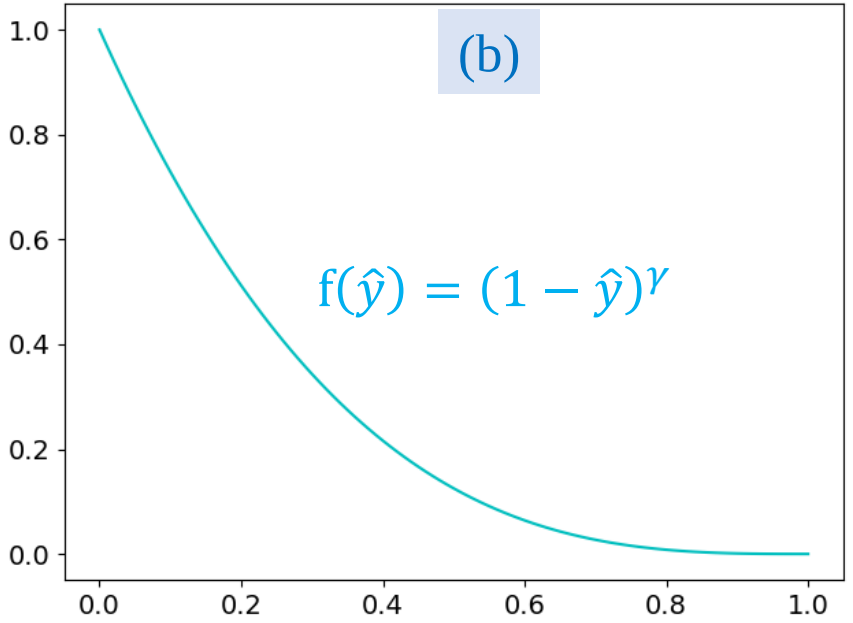
where k approaches 0

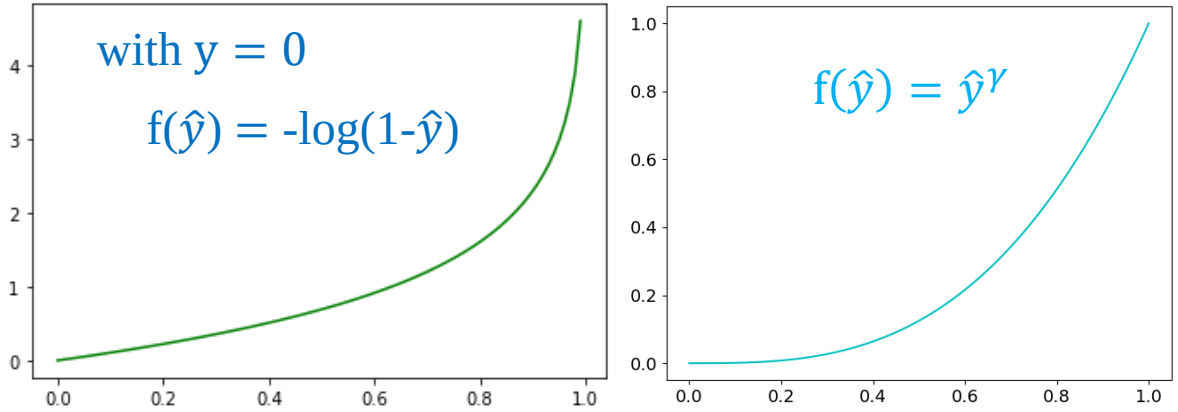
→ $f(\hat{y}) * k$ reduces significantly

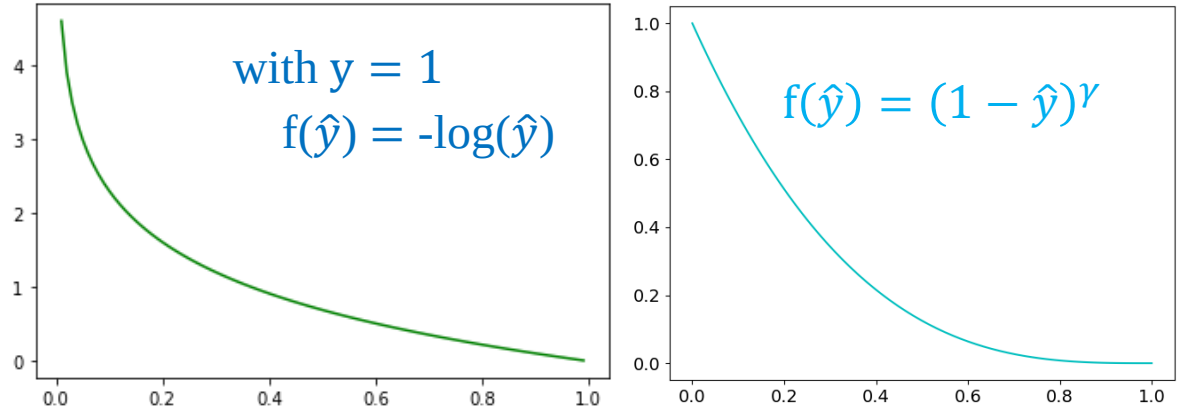
Reducing significantly for the correct part

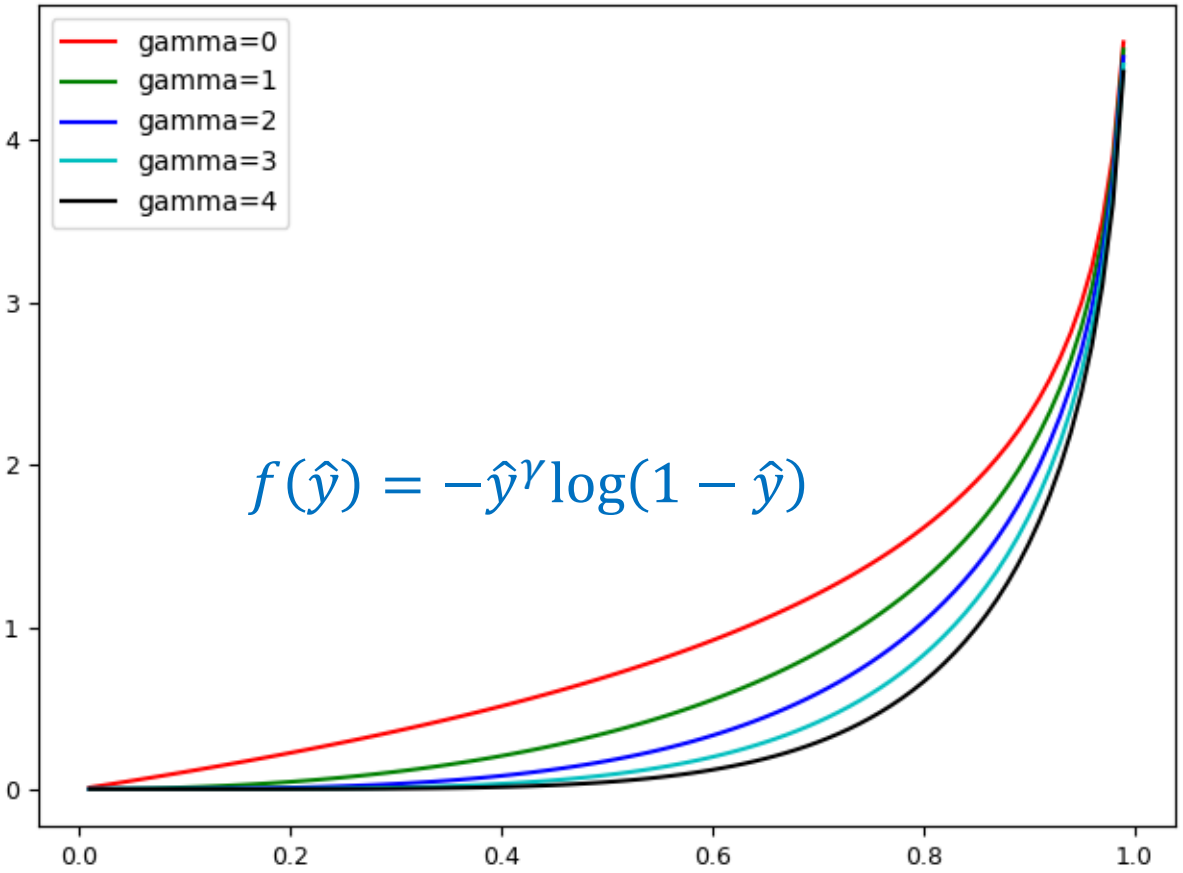
Reducing slightly for the incorrect part

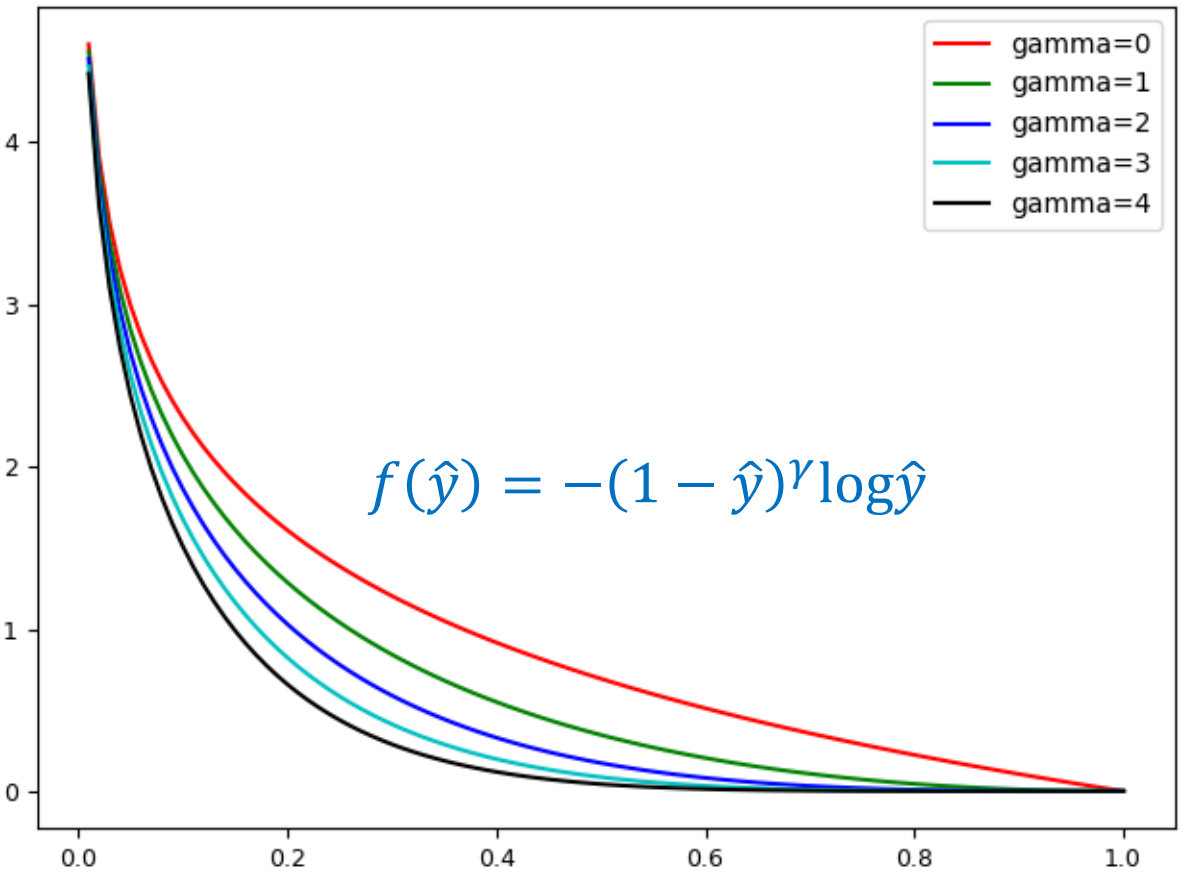





with y = 0
f(ŷ) = -log(1-ŷ)
f(ŷ) = ŷ^γ


with y = 1
f(ŷ) = -log(ŷ)
f(ŷ) = (1 − ŷ)^γ

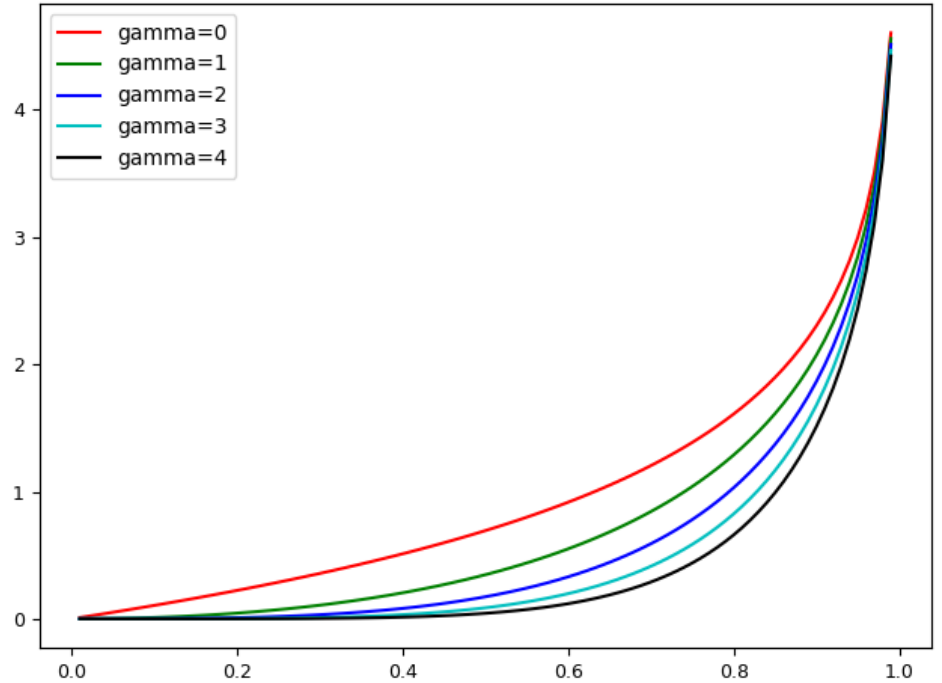

gamma=0
gamma=1
gamma=2
gamma=3
gamma=4
f(ŷ) = −ŷ^γ log(1 − ŷ)

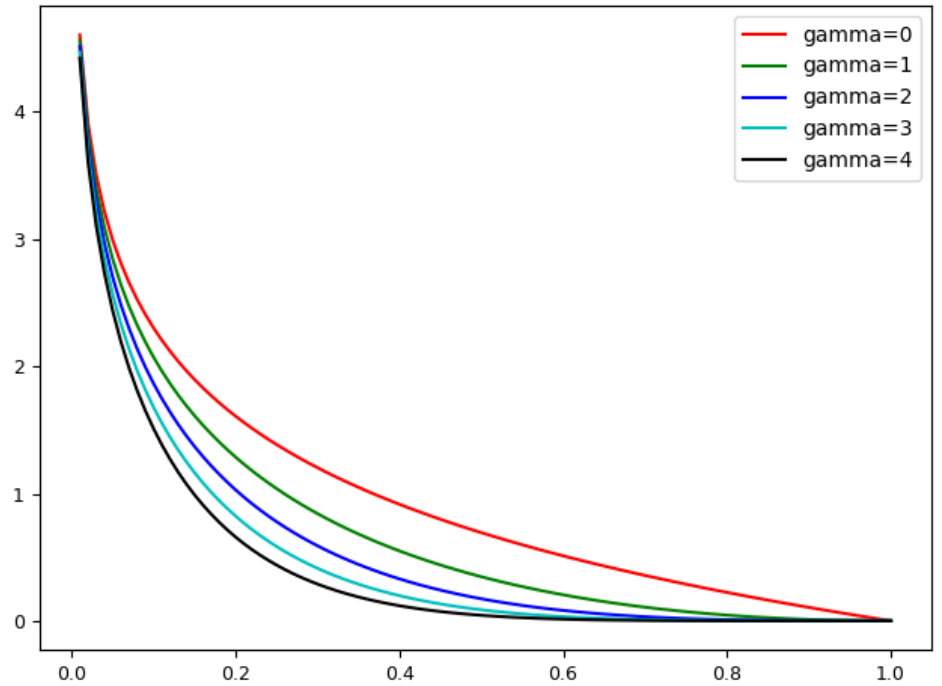

gamma=0
gamma=1
gamma=2
gamma=3
gamma=4
f(ŷ) = −(1 − ŷ)^γ logŷ

Applying to our Problem

$$L(y, \hat{y}, \gamma) = -y(1-\hat{y})^{\gamma}\log\hat{y} - (1-y)\hat{y}^{\gamma}\log(1-\hat{y})$$

Input	Output	Label	Gamma=0	Gamma=1	Gamma=2	Gamma=3	Gamma=4
…	0.7	0	1.204	0.842	0.589	0.412	0.289
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.7	1	0.356	0.107	0.032	0.009	0.002
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.9	1	0.105	0.011	0.001	0.0001	0.00001
…	0.9	1	0.105	0.011	0.001	0.0001	0.00001
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.7	1	0.356	0.107	0.032	0.009	0.002
$loss_{y=1}$:			2.039	0.458	0.111	0.028	0.007



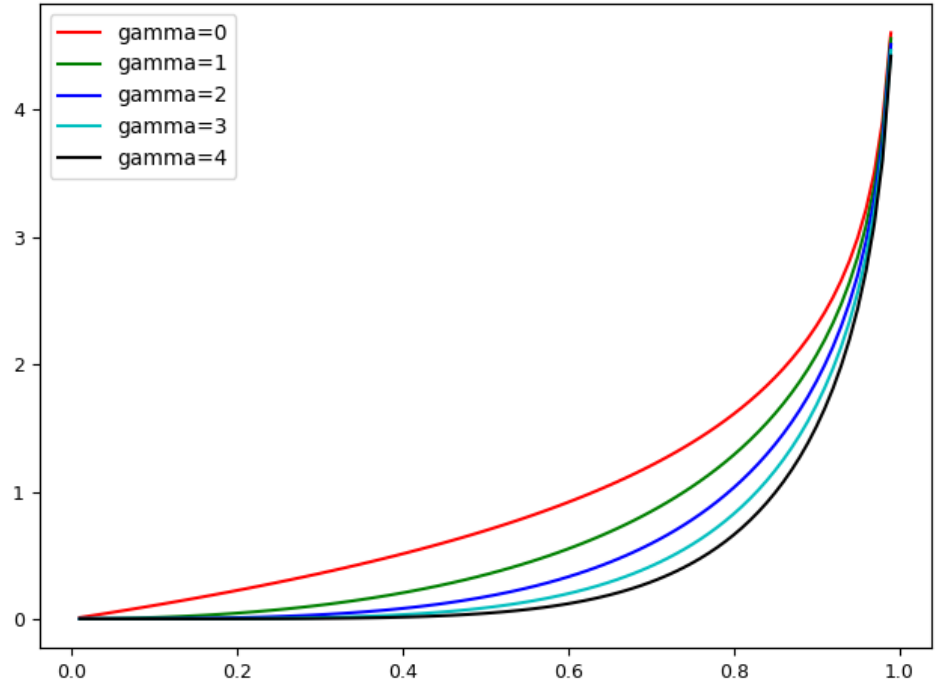


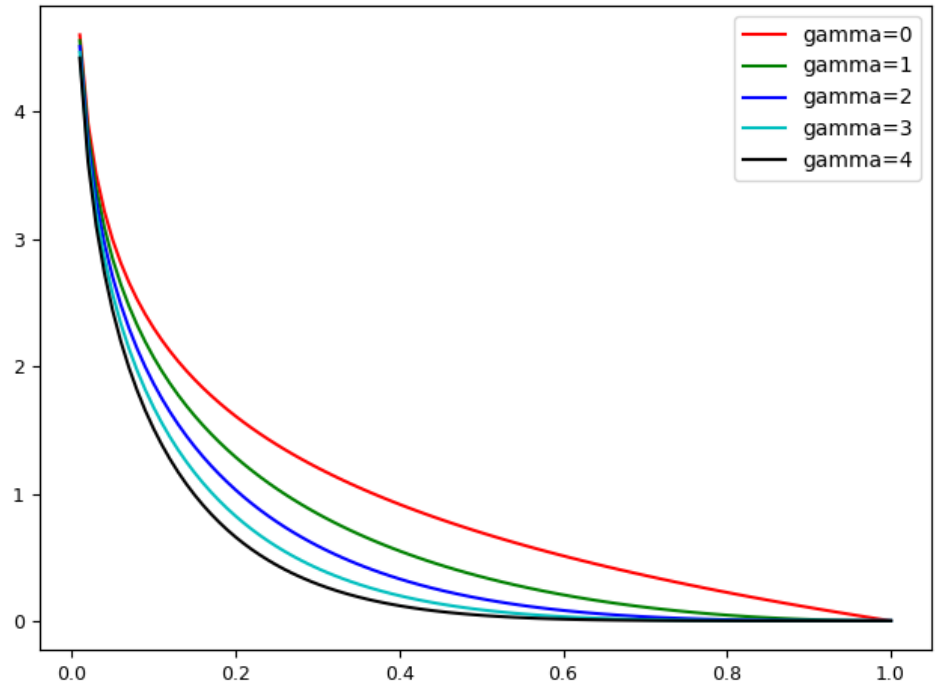
Focal Loss for Dense Object Detection, ICCV, 2017.

Combine with Class Weight

$$L(.) = -\alpha_1 y(1-\hat{y})^{\gamma}\log\hat{y} - \alpha_2(1-y)\hat{y}^{\gamma}\log(1-\hat{y})$$

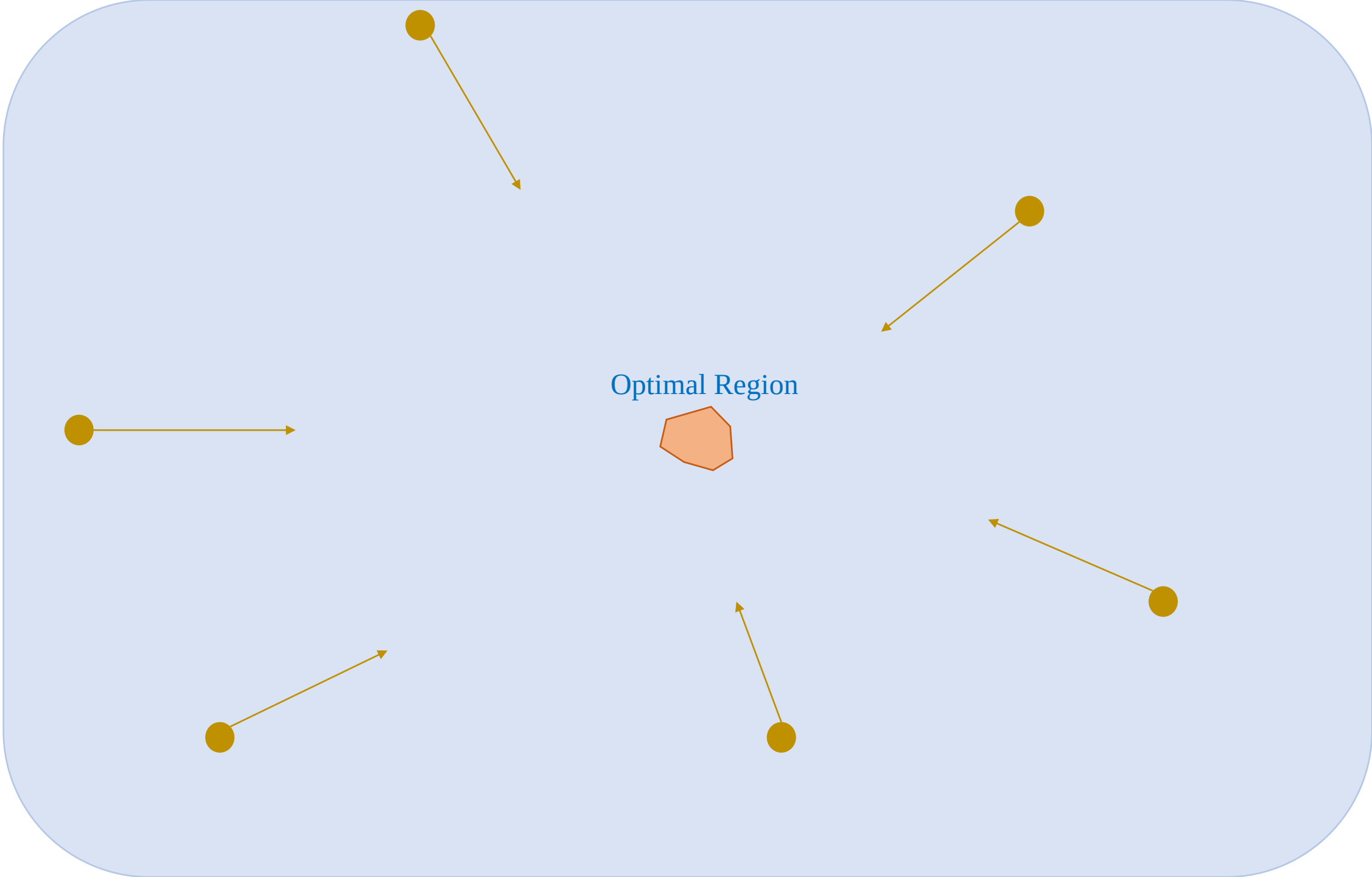
Input	Output	Label	Gamma=0	Gamma=1	Gamma=2	Gamma=3	Gamma=4
…	0.7	0	1.204	0.842	0.589	0.412	0.289
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.7	1	0.356	0.107	0.032	0.009	0.002
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.9	1	0.105	0.011	0.001	0.0001	0.00001
…	0.9	1	0.105	0.011	0.001	0.0001	0.00001
…	0.8	1	0.223	0.044	0.008	0.001	0.0003
…	0.7	1	0.356	0.107	0.032	0.009	0.002
$loss_{y=1}$:			2.039	0.458	0.111	0.028	0.007



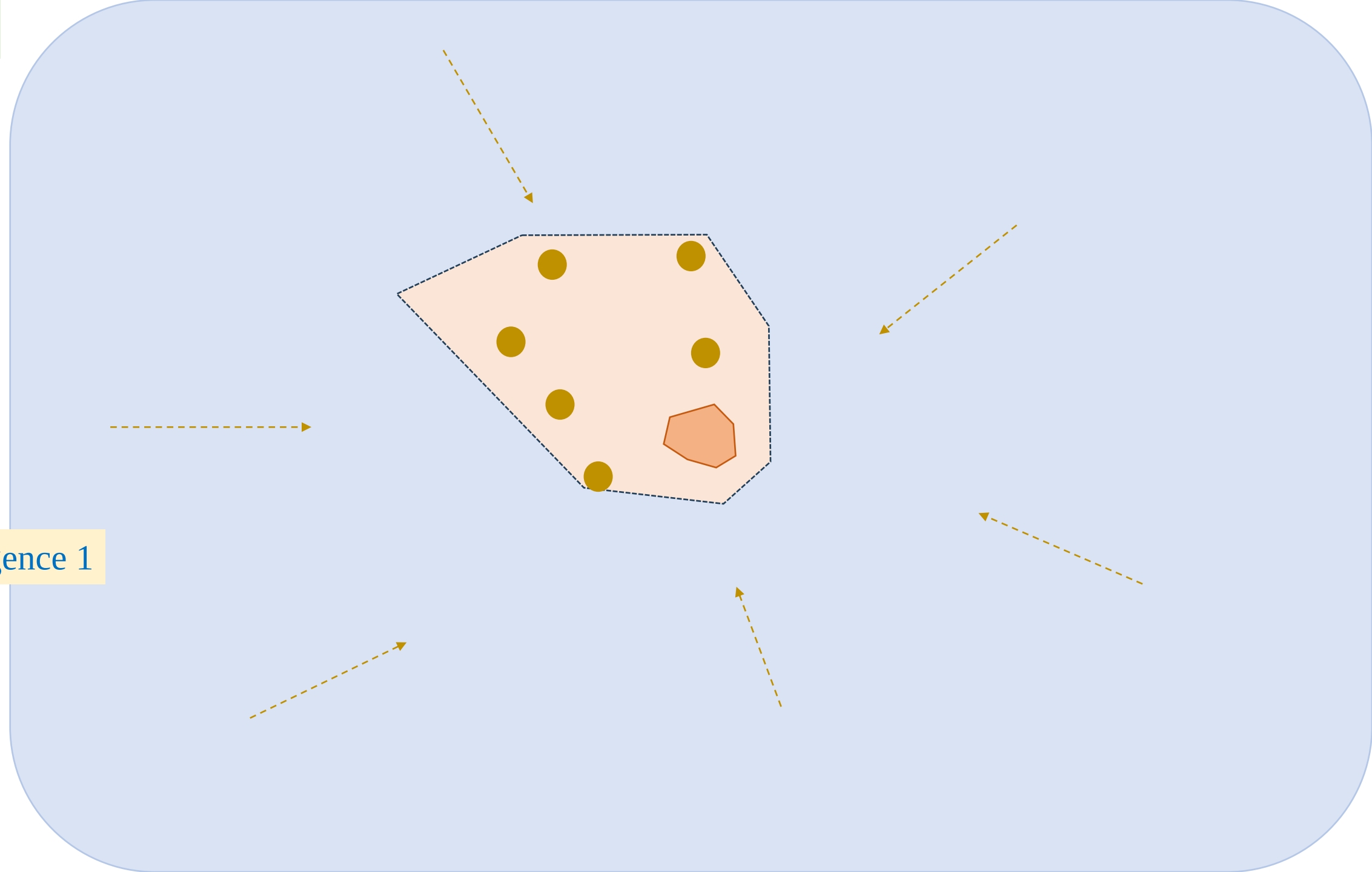


Focal Loss for Dense Object Detection, ICCV, 2017.

solution 5



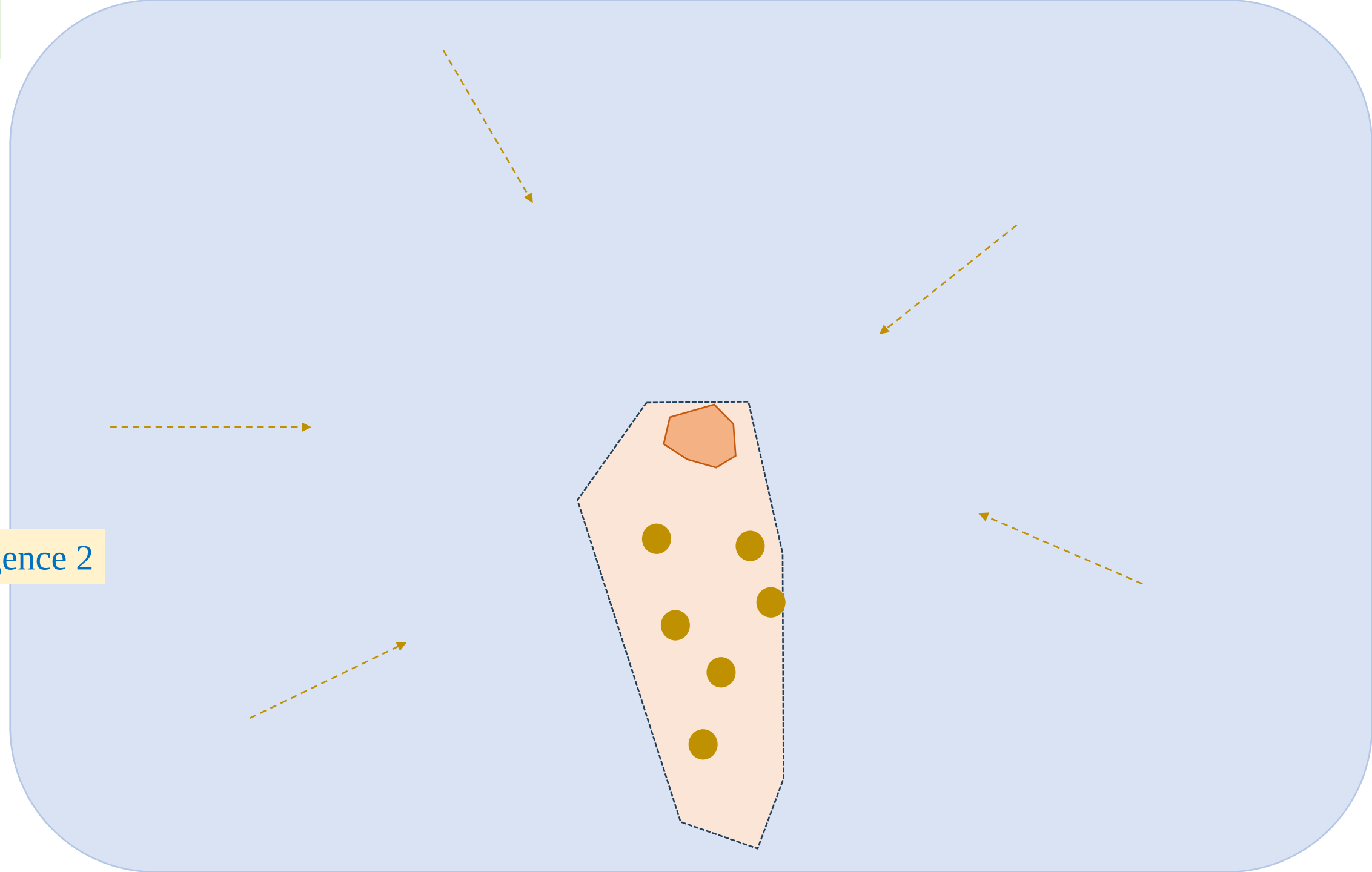
solution 5



convergence 1

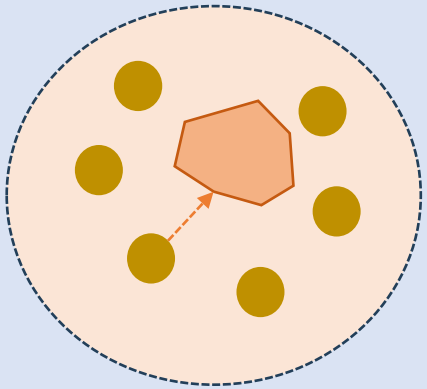
solution 5

convergence 2

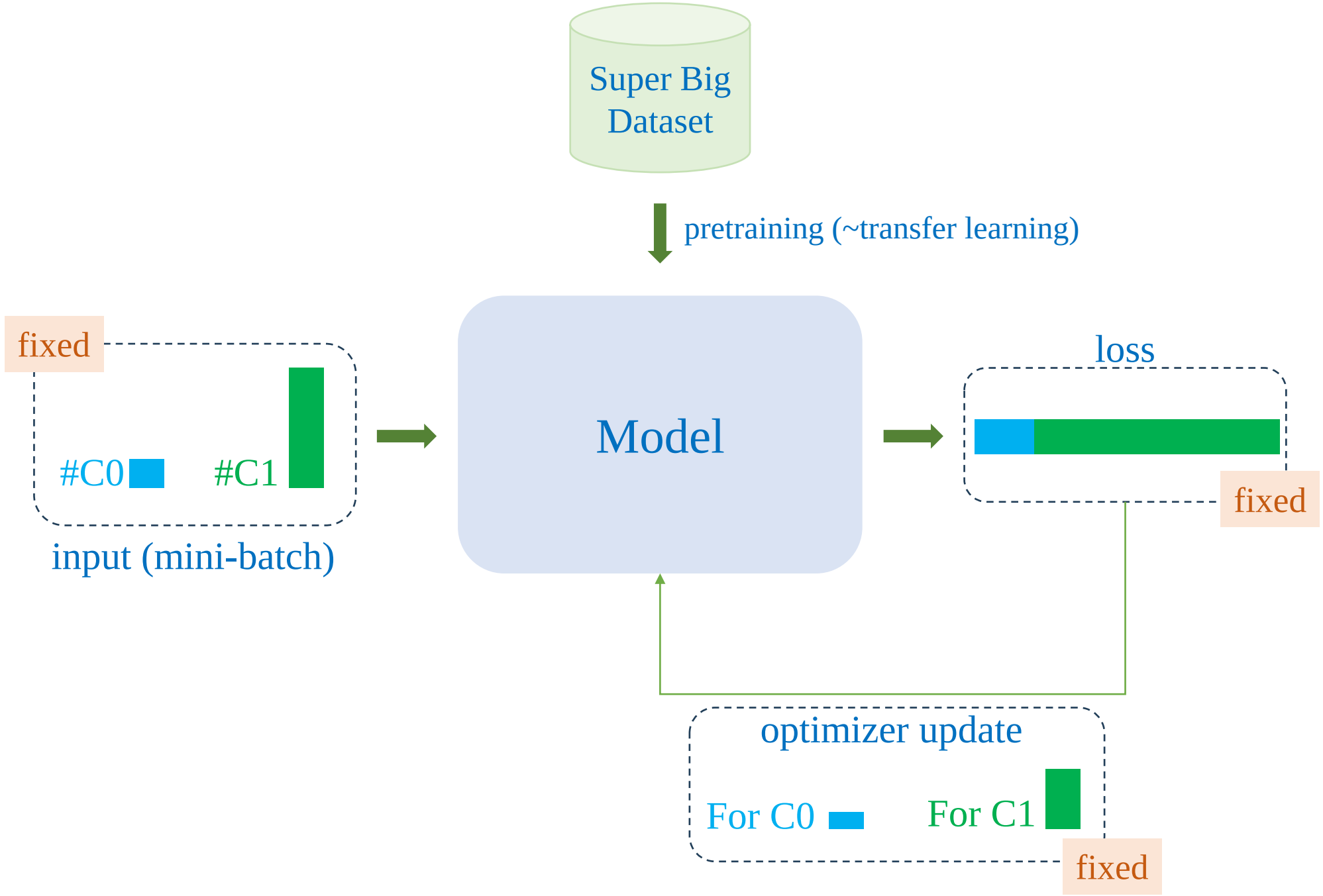


solution 5

Using Pretrained Model



solution 5

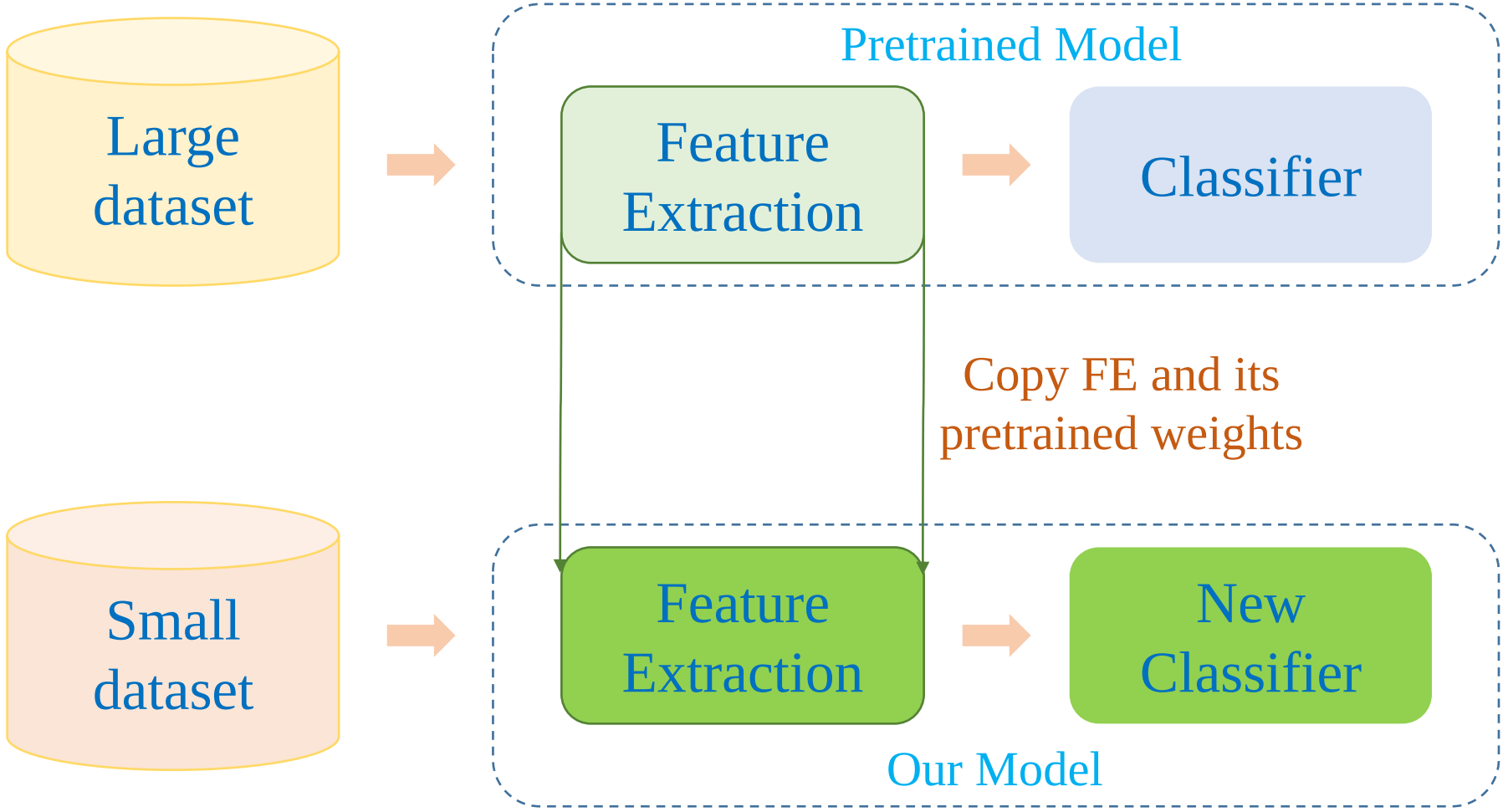

Super Big Dataset
pretraining (~transfer learning)
fixed
#C0
#C1
input (mini-batch)
Model
loss
fixed
optimizer update
For C0
For C1
fixed

Exploitation of Pretrained Models

❖ As Initialization

Will be trained with the small dataset

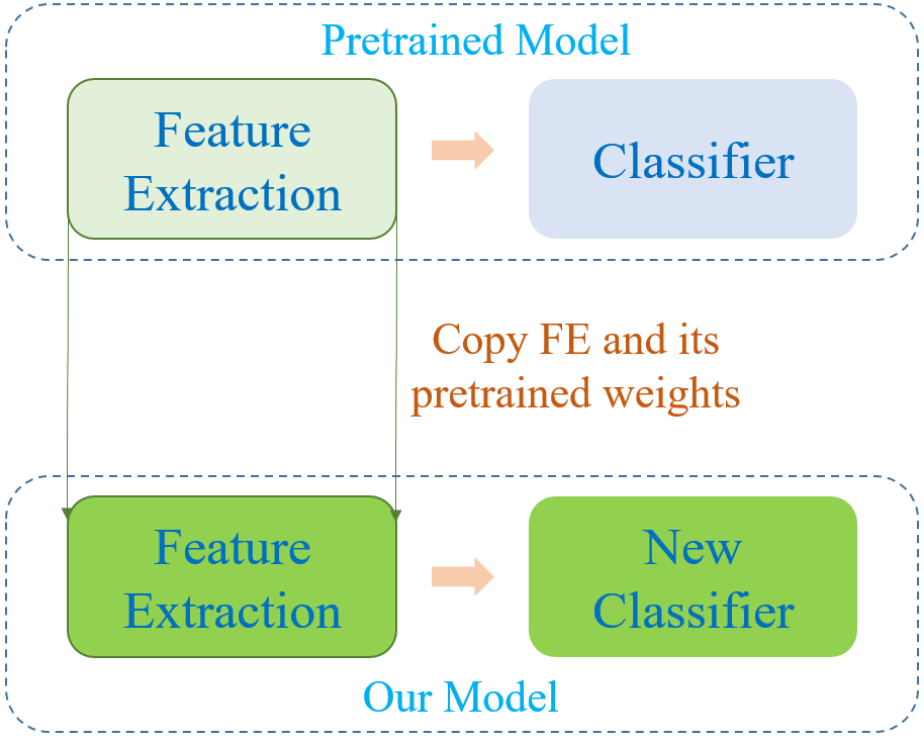
ImageNet dataset
(1.2 million images
of 1000 categories)



Train the whole new model using a small dataset

Exploitation of Pretrained Models

❖ **Use the pretrained weights as an initialization**

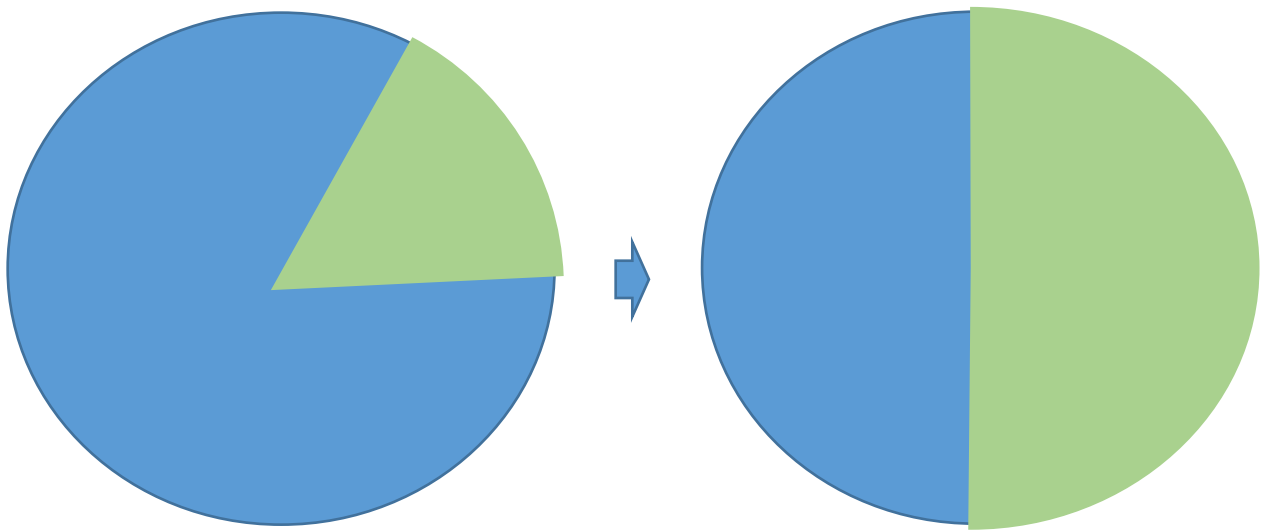



```python
# Load the pretrained VGG16 model
vgg16 = models.vgg16(weights=models.VGG16_Weights.DEFAULT)
f_extractor = vgg16.features

model = nn.Sequential(f_extractor,
                      nn.Flatten(),
                      nn.Dropout(0.3),
                      nn.Linear(512*7*7, 512),
                      nn.ReLU(),
                      nn.Dropout(0.3),
                      nn.Linear(512, 2))
```

```python
import torchvision.models as models
import torch.nn as nn

# Load Resnet18 model and pre-train weight
model = models.resnet18(weights="IMAGENET1K_V1")
model.fc = nn.Linear(512, 2, bias=True)
```

Imbalanced Data

❖ Approach 2: Loss Functions



Class weight

$$w_c = \frac{N}{2N_c}$$

Pay more attention to samples from an under-represented class

A higher loss → higher optimization

Focal loss

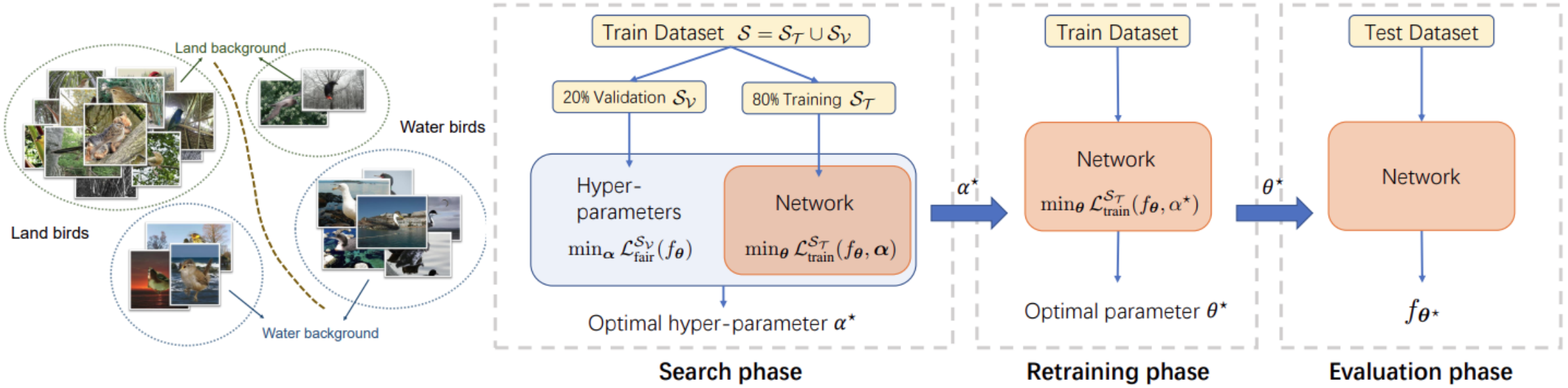
$$\mathrm{FL}(\mathrm{p_t}) = -\alpha_t(1-p_t)^{\gamma}\log(\mathrm{p_t})$$

γ helps the loss funcion focus on 'hard' samples

α_t balances losses using a number of samples in a class

Imbalanced Data

❖ Solution 6



https://arxiv.org/pdf/2201.01212.pdf

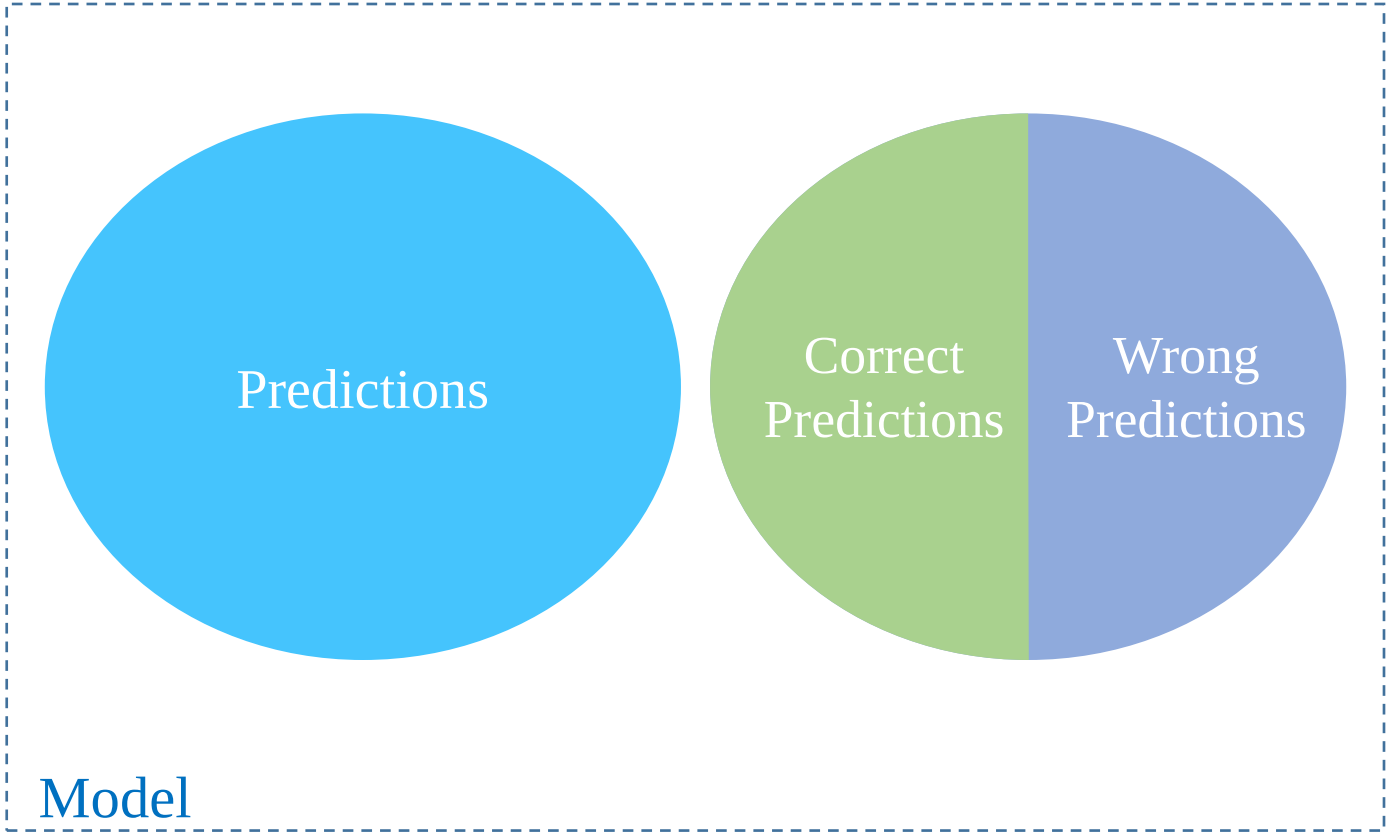
AutoBalance: Optimized Loss Functions for Imbalanced Data, 2022

Outline

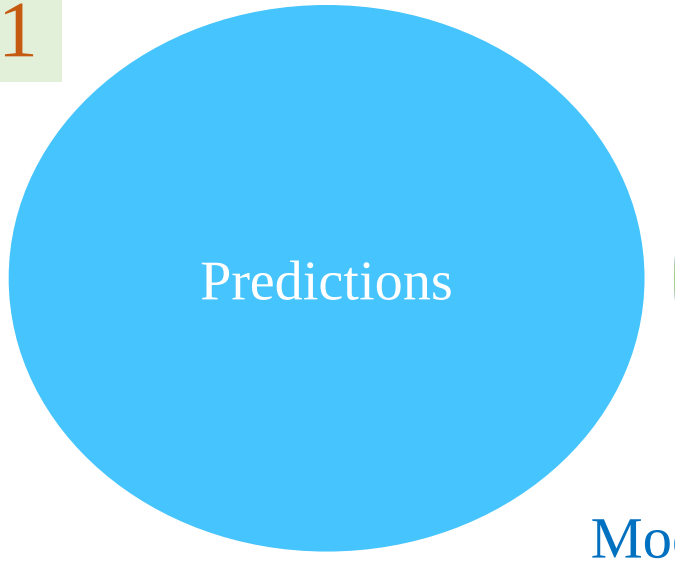
- Introduction
- Examples and Discussion
- Metrics
- Case Study

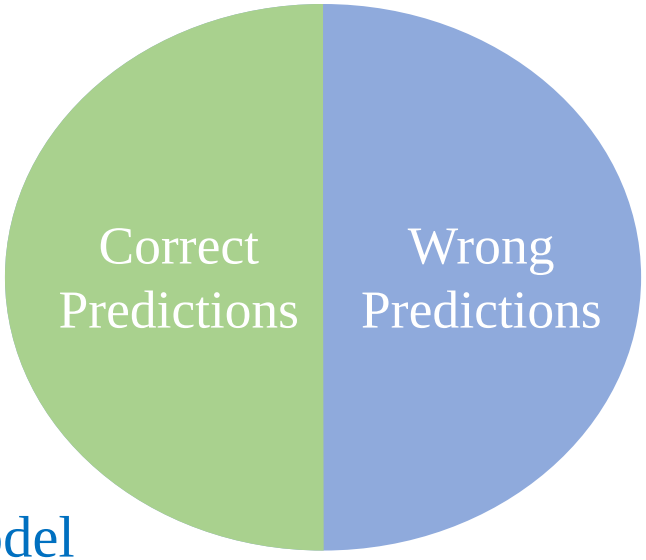
Metrics

❖ Confusion matrix



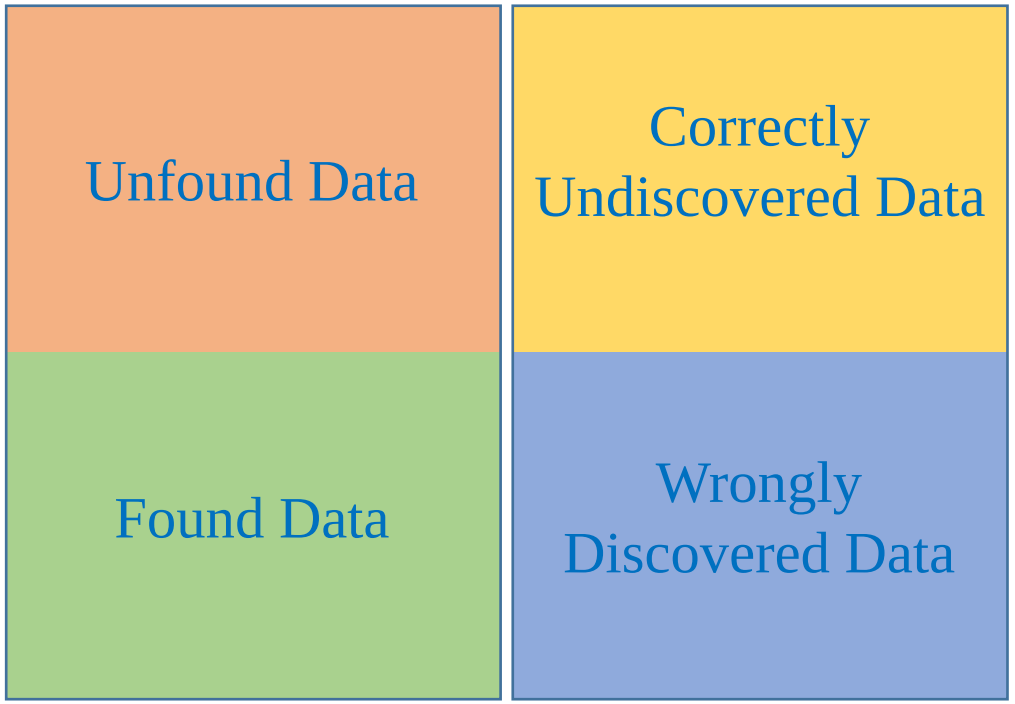
1

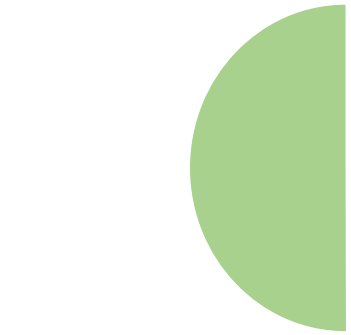




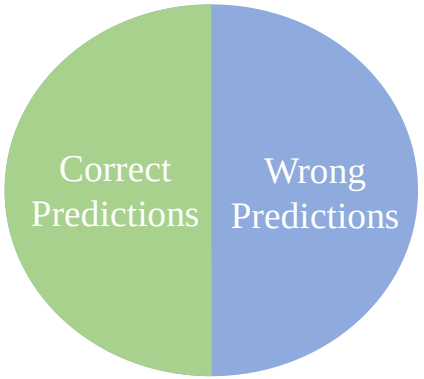
Model

2

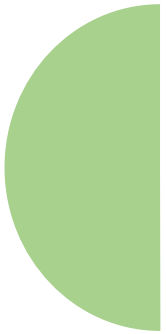




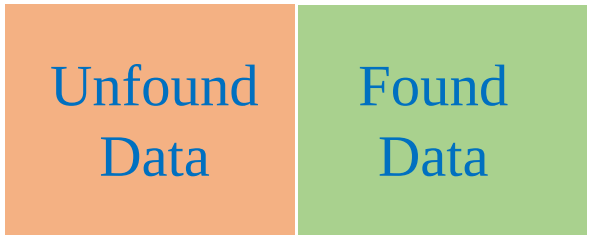
Precision



Precision=1?



Recall



Recall=1?

Metrics

❖ Precision and recall

Label

Result	Prediction

True Positive (TP): A correct positive detection

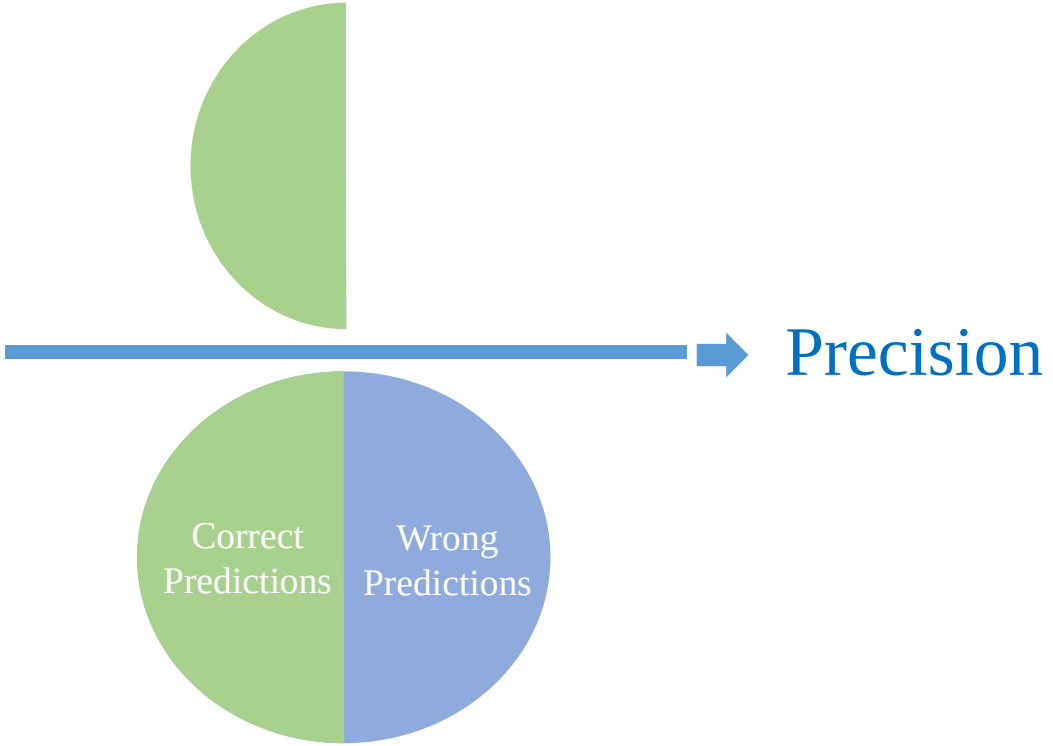
False Positive (FP): A wrong positive detection

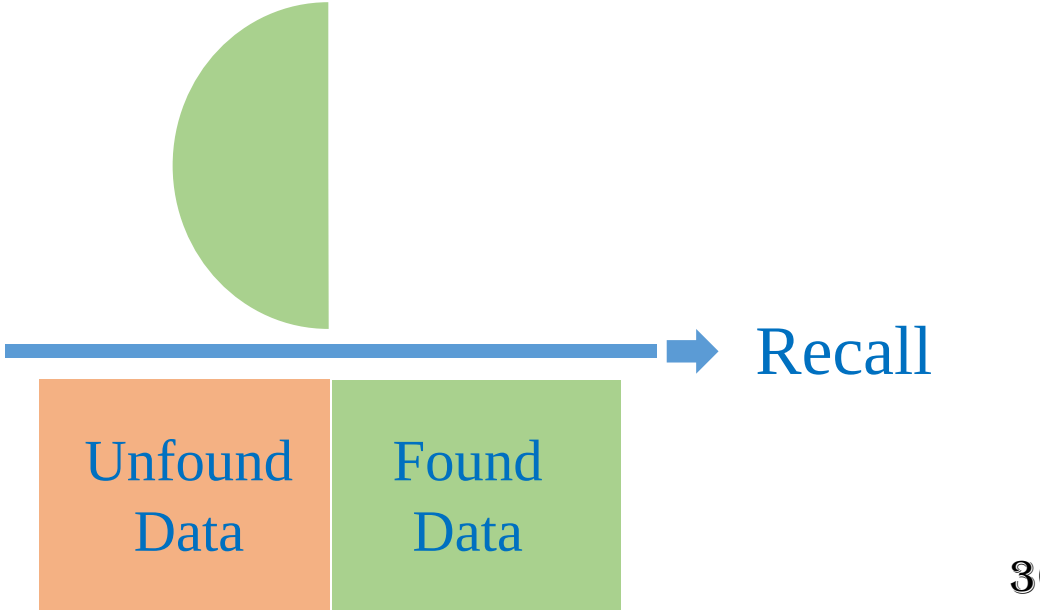
False Negative (FN): A wrong negative detection

True Negative (TN): A correct negative detection

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Số dự đoán chính xác}}{\text{Tổng số lần dự đoán positive}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Số dự đoán chính xác}}{\text{Tổng số ground truth cho positive}}$$





Metrics

❖ Quiz

Fill TP, FP, TN, FN into appropriate cells

	Actual: NEGATIVE	**Actual**: POSITIVE
Predicted: NEGATIVE		
Predicted: POSITIVE		

Metrics

❖ Accuracy

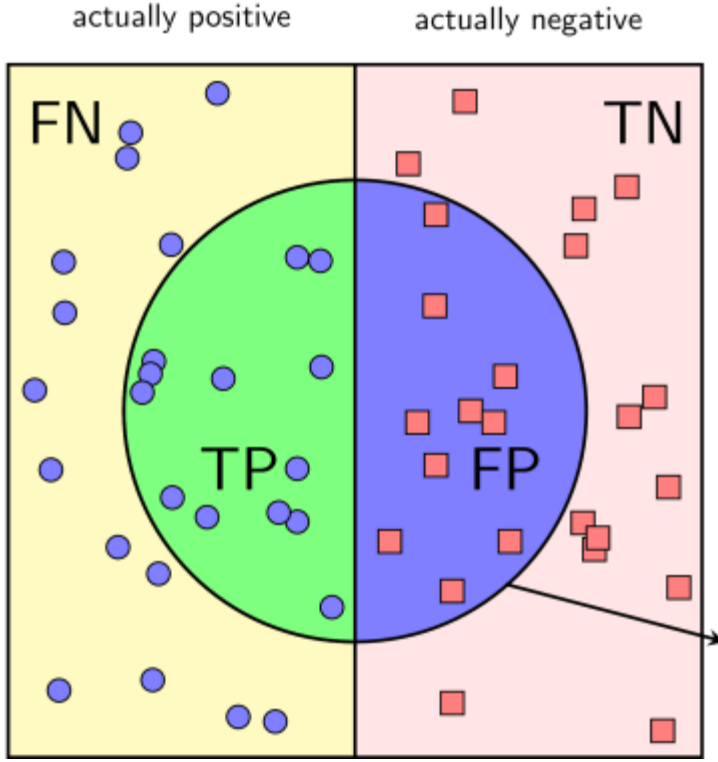
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

❖ Precision

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ positive\ decisions}$$

❖ Recall

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ positive\ samples}$$



$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} =$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} =$$

Metrics

❖ Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

❖ Precision

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ positive\ decisions}$$

❖ Recall

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ positive\ samples}$$

n = 165	**Predicted:** NEGATIVE	**Predicted:** POSITIVE	
Actual: NEGATIVE	TN=50	FP=10	60
Actual: POSITIVE	FN=10	TP=95	105
	60	105	

Accuracy = 145/165

Precision: When it predicts yes, how often is it correct?

TP/predicted yes = 95/105 = 0.9

Recall: When it's actually yes, how often does it predict yes?

TP/actual yes = 95/105 = 0.90

Metrics

❖ Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

❖ Precision

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ positive\ decisions}$$

❖ Recall

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ positive\ samples}$$

n = 165	**Predicted:** NEGATIVE	**Predicted:** POSITIVE	
Actual: NEGATIVE	TN=60	FP=0	60
Actual: POSITIVE	FN=20	TP=85	105
	80	85	

Accuracy = 145/165

Precision: When it predicts yes, how often is it correct?

TP/predicted yes = 85/85 = 1.0

Recall: When it's actually yes, how often does it predict yes?

TP/actual yes = 85/105 = 0.81

Metrics

❖ Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

❖ Precision

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ positive\ decisions}$$

❖ Recall

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ positive\ samples}$$

n = 165	**Predicted:** NEGATIVE	**Predicted:** POSITIVE	
Actual: NEGATIVE	TN=40	FP=20	60
Actual: POSITIVE	FN=0	TP=105	105
	60	105	

Accuracy = 145/165

Precision: When it predicts yes, how often is it correct?

TP/predicted yes = 105/125 = 0.84

Recall: When it's actually yes, how often does it predict yes?

TP/actual yes = 105/105 = 1.0

AI VIETNAM

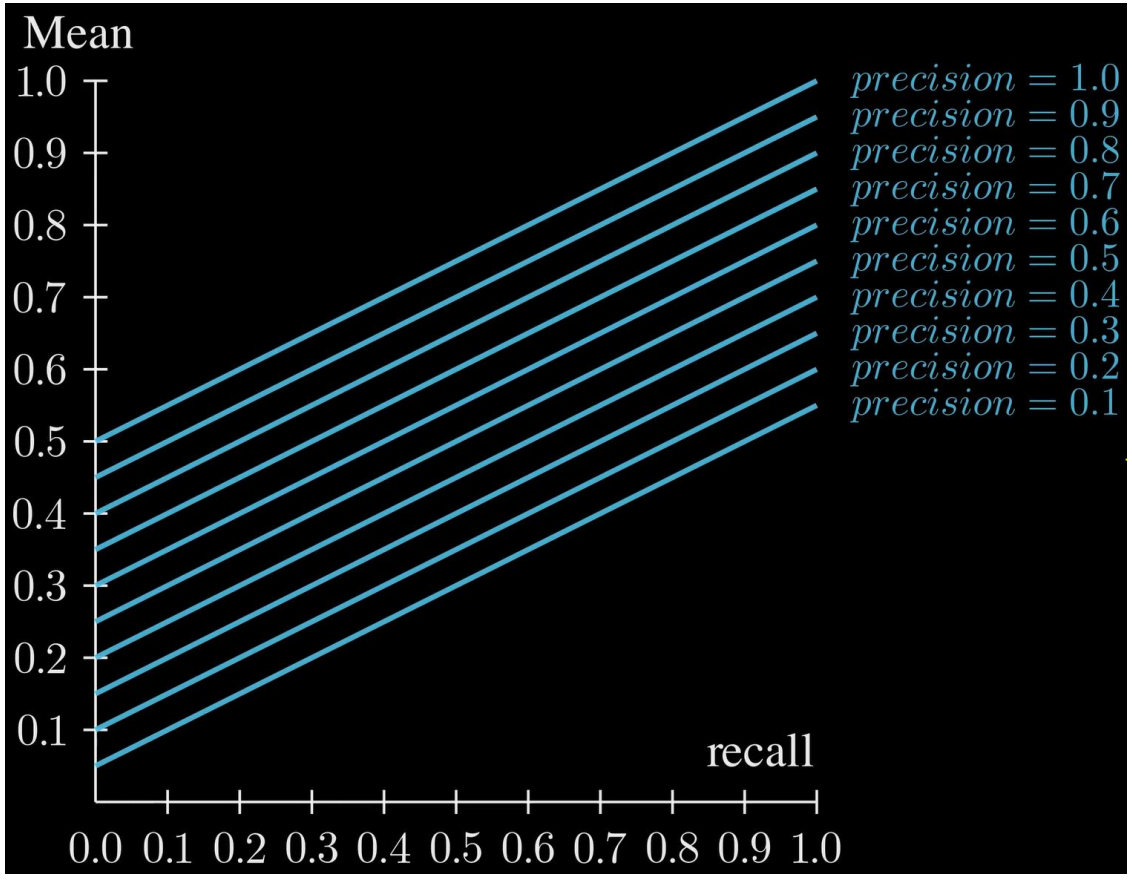
AI VIETNAM

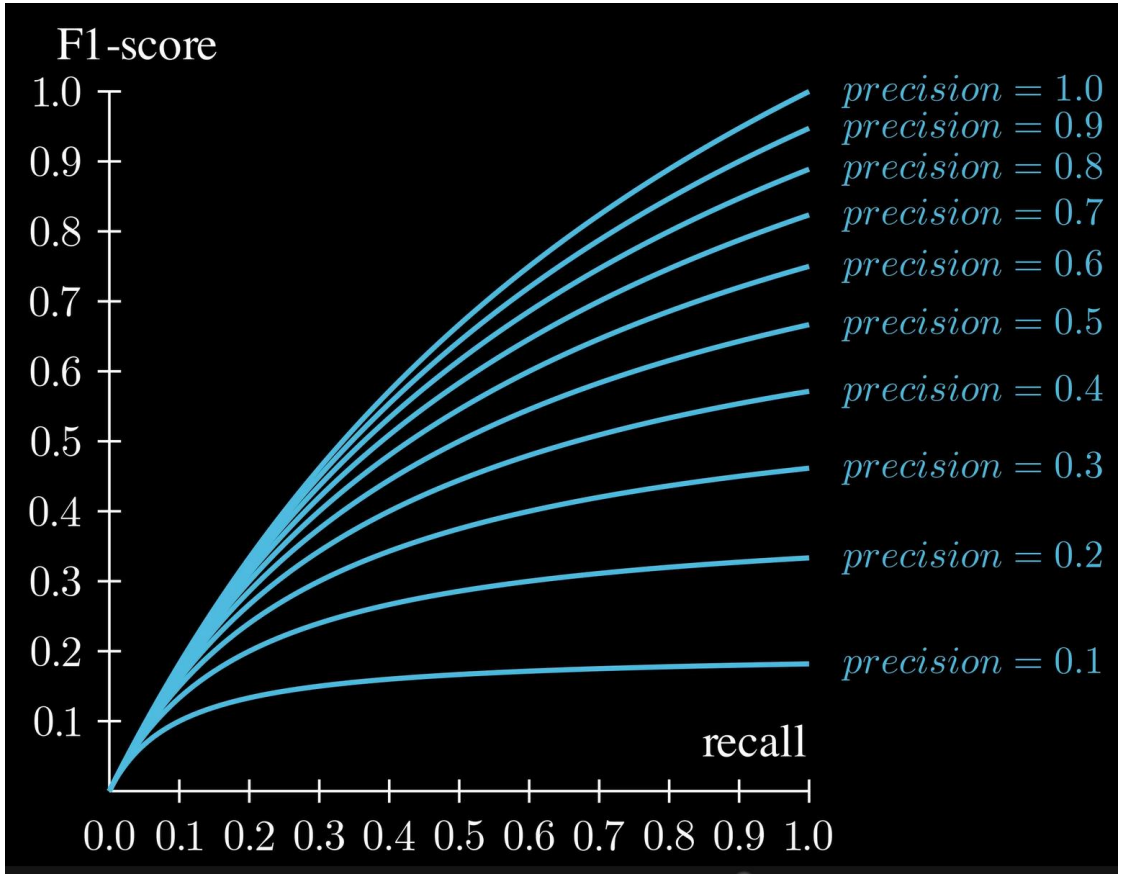
Metrics

❖ Combine precision and recall

$$F_1 = \frac{\text{precision} + \text{recall}}{2}$$

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}}$$





Metrics

❖ F1 Score

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

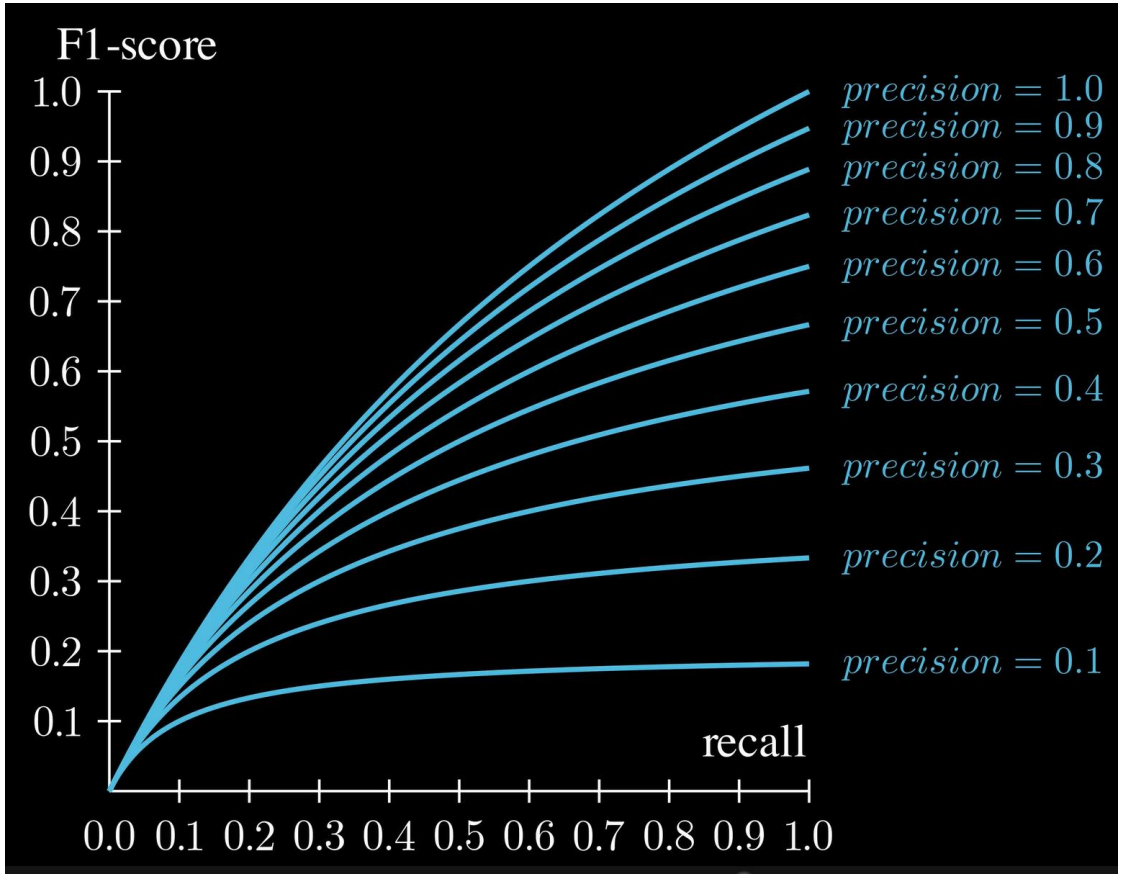
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\frac{2}{\text{F}_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}}$$

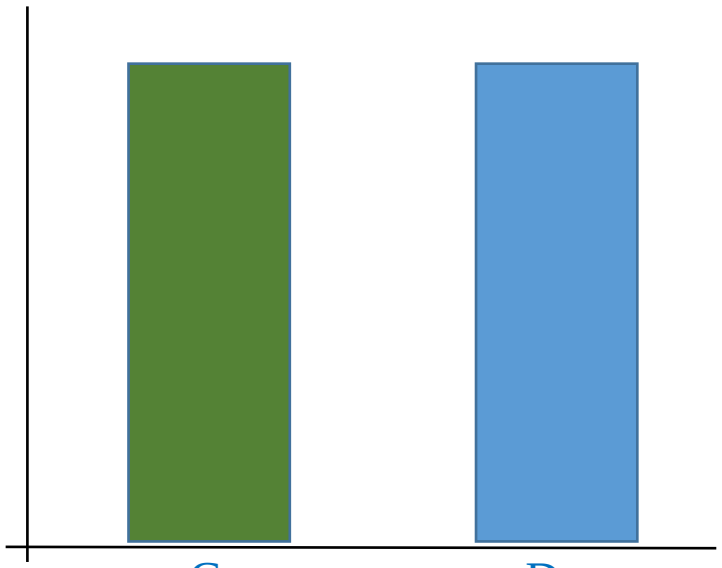
$$\text{F}_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precion} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Precision	Recall	F1
1	1	1
0.1	0.1	0.1
0.5	0.5	0.5
1	0.1	0.182
0.3	0.8	0.36



Example

❖ Cat-Dog dataset



Cat is of the positive class

Validation data (3000 samples)

Correct prediction

$$\#_{cat} = 821$$
$$\#_{dog} = 1489$$

n=3000	Predicted: NEGATIVE	Predicted: POSITIVE	
Actual: NEGATIVE	TN=1489	FP=11	1500
Actual: POSITIVE	FN=679	TP=821	1500
	2168	832	

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{821}{821 + 679} \approx 0.547$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{821}{821 + 11} \approx 0.987$$

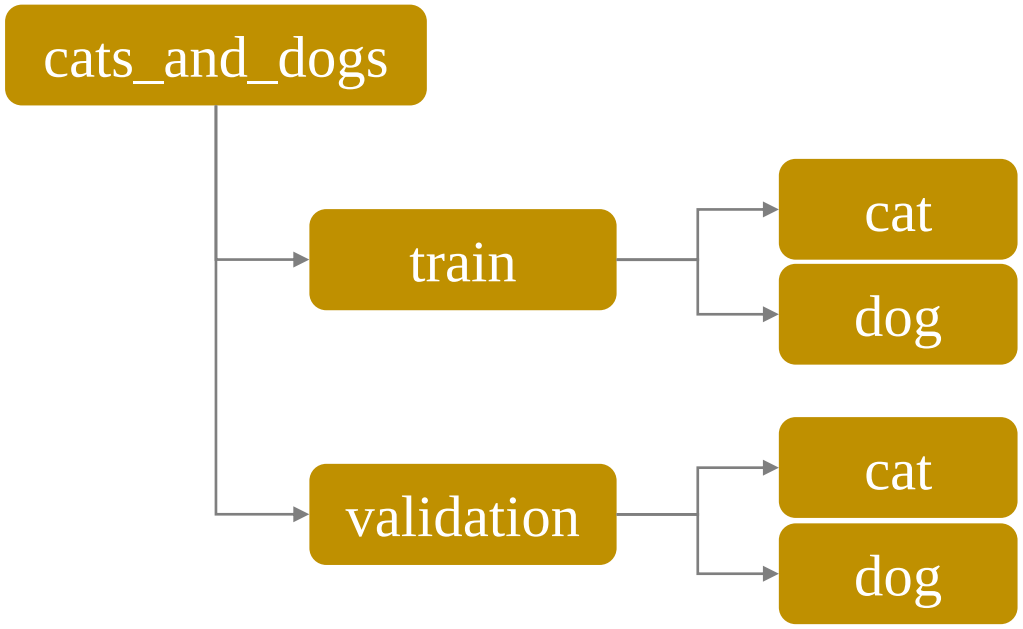
$$\text{F1} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{2 * 0.547 * 0.987}{0.547 * 0.987} \approx 0.704$$

Outline

- Introduction
- Examples and Discussion
- Metrics
- Case Study

Experiments

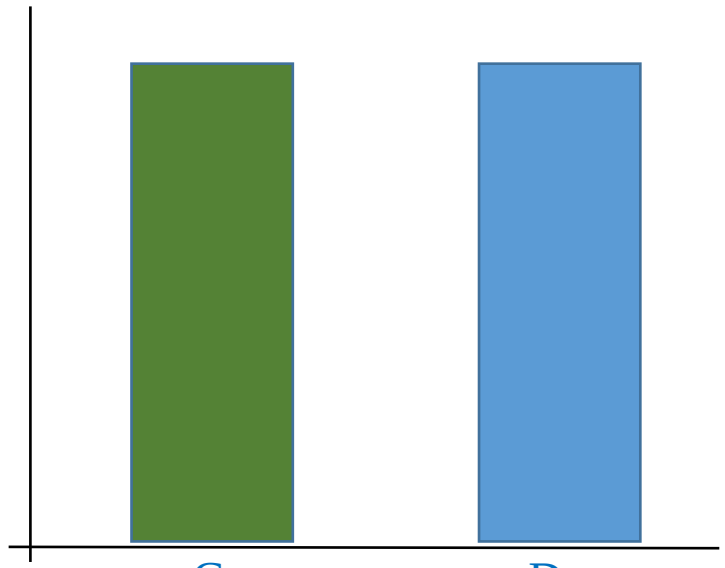
❖ Cat-Dog dataset



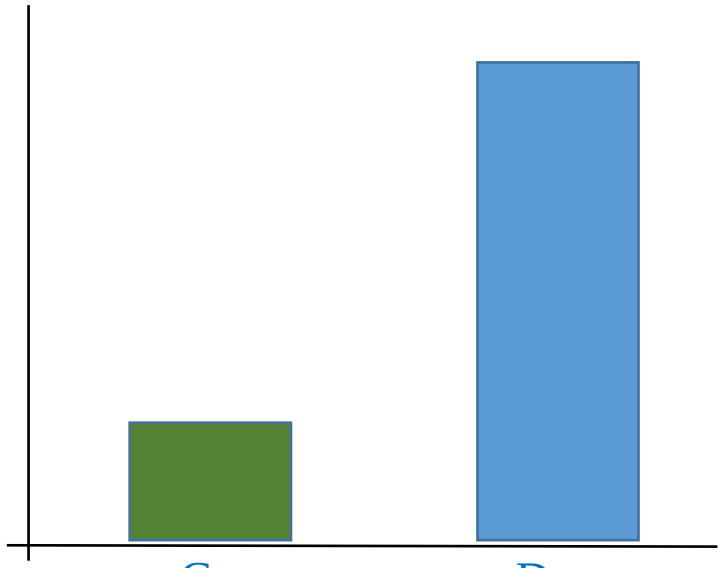
Experiments

❖ Cat-Dog dataset

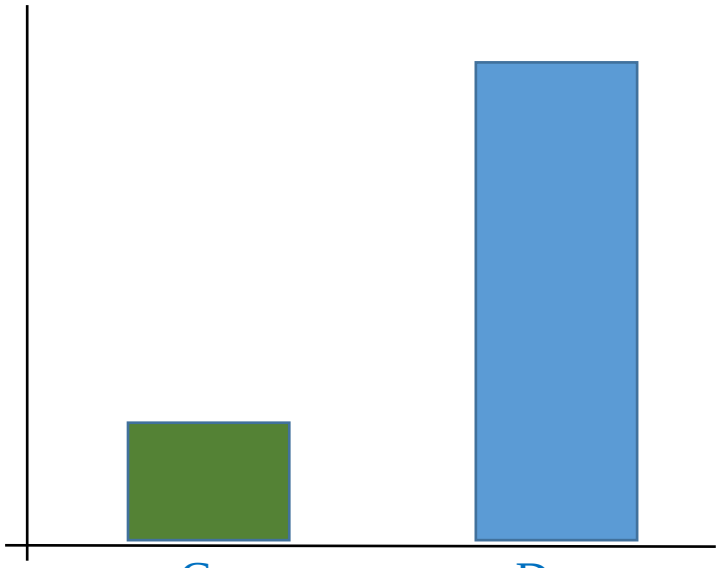
Validation data (3000 samples)



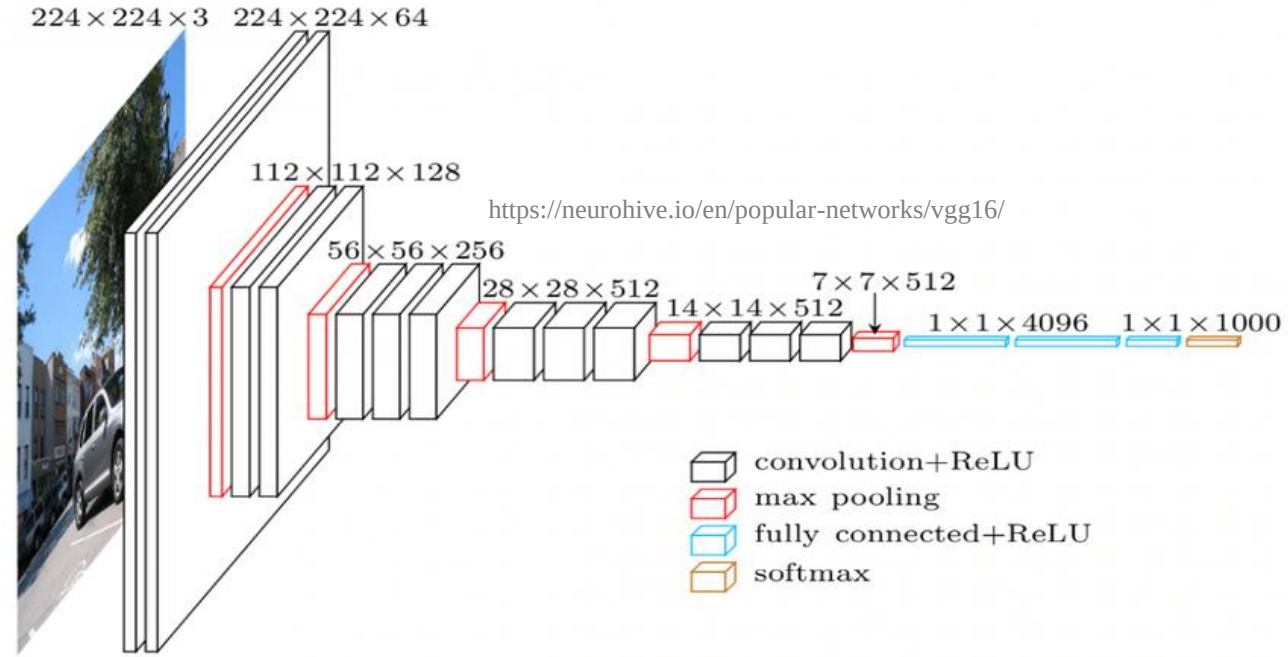
Balanced Data



Imbalanced Data 1



Imbalanced Data 2



https://neurohive.io/en/popular-networks/vgg16/

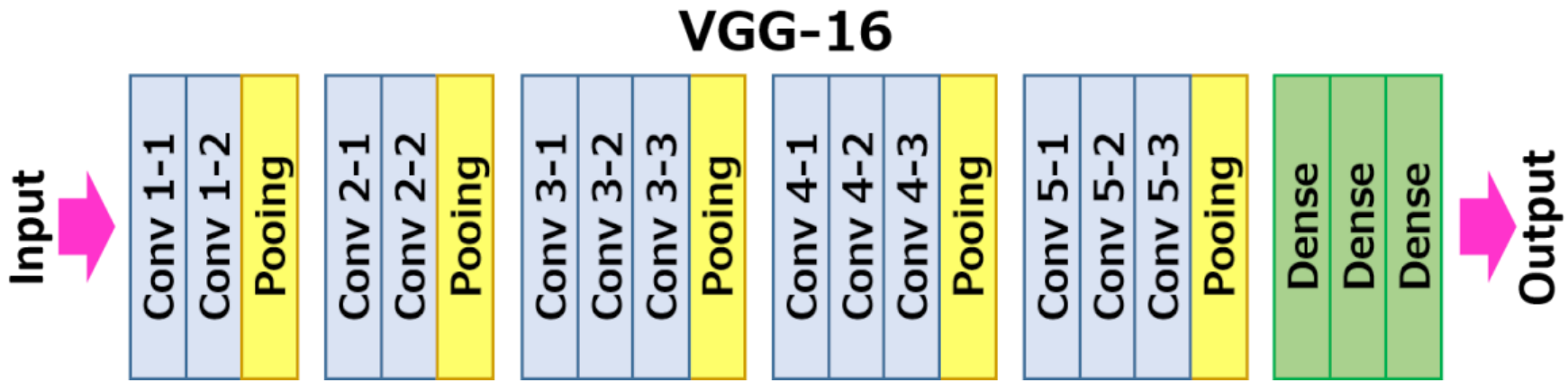
(3x3) Convolution padding='same' stride=1 + ReLU

(2x2) max pooling

Dense Layer + ReLU (4096)

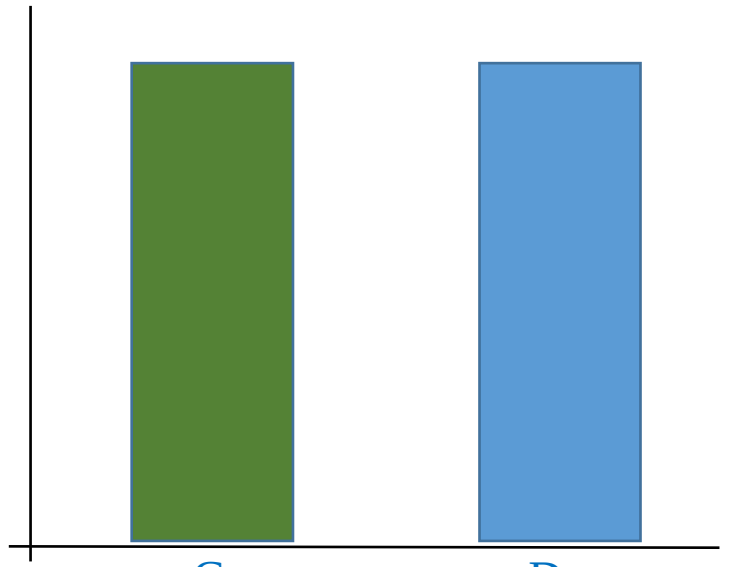
Dense Layer + Softmax (1000)

VGG16



Experiments

❖ Cat-Dog dataset: Results from the TF codes



Balanced Data

Validation data (3000 samples)

Balanced Loss	Correct prediction
$L_b = L_c + L_d$	$\#_{cat} = 1445$ $\#_{dog} = 1436$ $F_1 = 0.96$

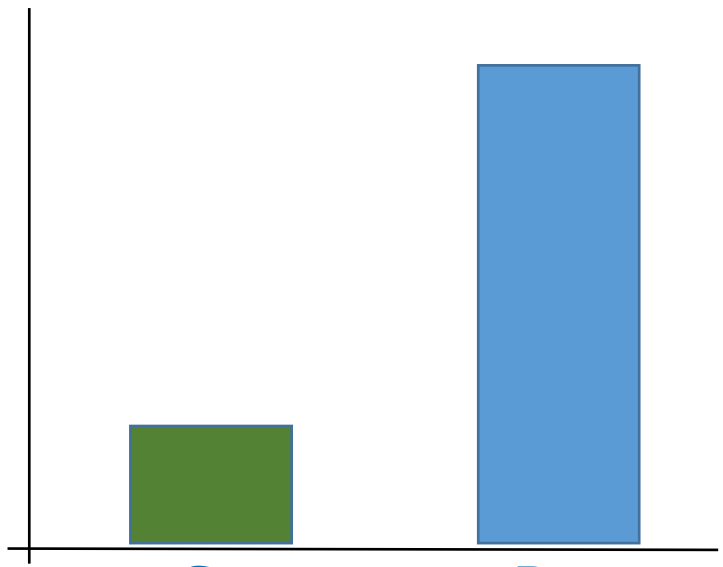
Imbalanced Loss 1	Correct prediction
$L_b = L_c + 100 \times L_d$	$\#_{cat} = 1076$ $\#_{dog} = 1499$ $F_1 = 0.835$

Imbalanced Loss 2	Correct prediction
$L_b = L_c + 1000 \times L_d$	$\#_{cat} = 670$ $\#_{dog} = 1498$ $F_1 = 0.617$

Experiments

❖ Cat-Dog dataset

Results from the TF codes



Imbalanced Data 1

Validation data (3000 samples)

Balanced Loss	Correct prediction
$L_b = L_c + L_d$	$\#_{cat} = 1082$ $\#_{dog} = 1483$ $F_1 = 0.833$

Imbalanced Loss	Correct prediction
$L_b = 6 \times L_c + 0.55 \times L_d$	$\#_{cat} = 1163$ $\#_{dog} = 1379$ $F_1 = 0.835$

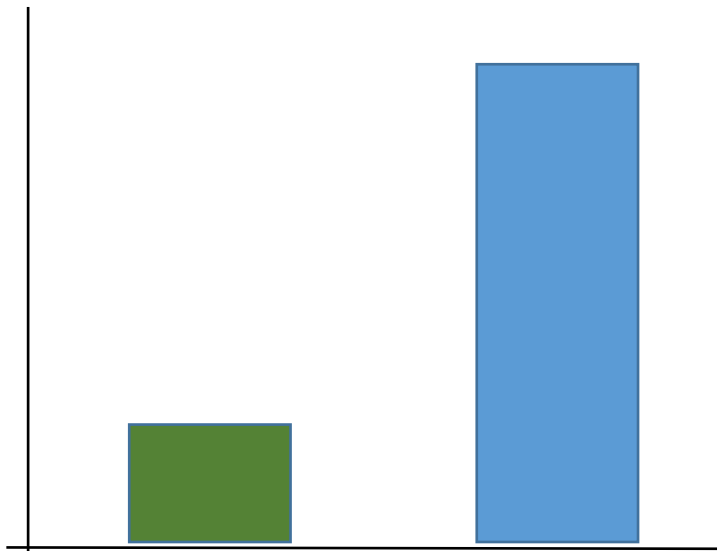
Oversampling Data	Focal loss
$\#_{cat} = 1167$ $\#_{dog} = 1438$ $F_1 = 0.855$	$\#_{cat} = 1210$ $\#_{dog} = 1447$ $F_1 = 0.876$

$\mathrm{FL}(\mathrm{p_t}) = -\alpha_t(1-p_t)^{\gamma}\log(\mathrm{p_t})$ $\gamma = 2.0$ $\alpha_t = 0.5$

Experiments

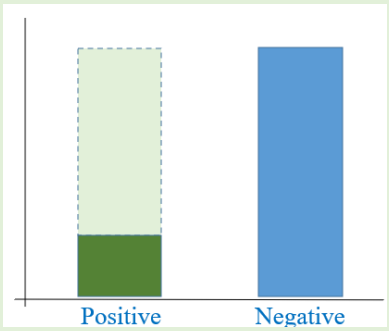
❖ Cat-Dog dataset

Results from the TF codes



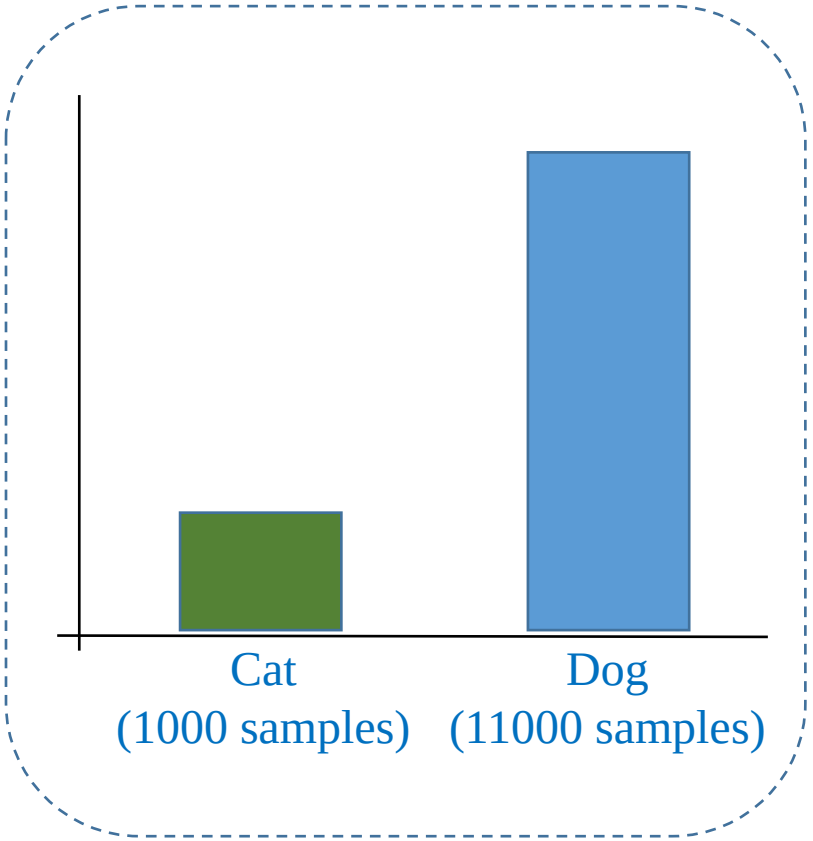
Imbalanced Data 2

Validation data (3000 samples)

Balanced Loss $L_b = L_c + L_d$	Correct prediction $\#_{cat} = 821$ $\#_{dog} = 1489$ $F_1 = 0.704$
Imbalanced Loss $L_b = 11.5 \times L_c + 0.52 \times L_d$	Correct prediction $\#_{cat} = 1123$ $\#_{dog} = 1309$ $F_1 = 0.798$
Oversampling 	Correct prediction $\#_{cat} = 1159$ $\#_{dog} = 1386$ $F_1 = 0.836$
Focal Loss $\mathrm{FL}(\mathrm{p_t}) = -\alpha_t(1-p_t)^{\gamma}\log(\mathrm{p_t})$ $\gamma = 2.0 \quad \alpha_t = 0.5$	Correct prediction $\#_{cat} = 1210$ $\#_{dog} = 1447$ $F_1 = 0.876$

Experiments

❖ Cat-Dog dataset: Using Resnet and Pytorch



Imbalanced Data 1

Normal cross-entropy

$F_1 = 0.67$

Class weight

$F_1 = 0.75$

Focal Loss

$F_1 = ???$

Using the pretrained model

$F_1 = 0.96$